MATLAB

MATLAB 完全学习手册



赵国生 主 编

于 翔 王 健 副主编



MATLAB完全学习手册

内容全面，覆盖面广： 详细介绍了数值计算、符号运算、图形图像、MATLAB工具箱以及Simulink仿真等方面的内容

由浅入深，图文并茂： 每章都从基础知识开始介绍，然后是实例分析，最后是习题练习，理论与实践紧密结合

案例丰富，讲解深刻： 在讲解过程中配以大量实例，使读者循序渐进地熟悉软件、学习软件、掌握软件



PPT课件+视频演示
+源代码+习题答案

清华大学出版社

MATLAB 完全学习手册

赵国生　于翔　王健　主编

清 华 大 学 出 版 社
北　京

内 容 简 介

目前，MATLAB 已发展成为国际公认的优秀数学应用软件之一，与 Mathematica、Maple 并称为三大数学软件，其在数值计算方面更是首屈一指。掌握了这一工具的使用将使日常学习和工作事半功倍。

本书对 MATLAB 进行了详细讲解，并配有大量实例，达到零起点入门和快速提高的目的。本书共分为 2 篇，前 7 章为基础篇，讲解有关 MATLAB 的基础知识，包括 MATLAB 的安装、卸载及系统功能的简述，MATLAB 的数值运算、符号运算和图形功能，M 文件编程、Simulink 框图仿真及图形用户界面等内容。第 8～11 章为进阶篇，第 8 章和第 9 章分别介绍了 MATLAB 的科学计算、S-函数的概念、原理和应用。第 10 章和第 11 章分别介绍了 MATLAB 工具箱及 MATLAB 外部接口。

本书内容丰富、全面，示例精巧，条理清晰、深入浅出、指导性强。在本书的 MATLAB 编程实现中，源程序详尽、清晰，注释丰富，而且通过实验验证了其正确性。通过章后的习题练习，不但可以帮助读者快速掌握本章理论，还可在编程中进一步熟练掌握 MATLAB 的高级编程技巧。

本书适合作为各大中专院校的理工科学生的专业教材，也可以作为读者自学的教程和各类科研技术人员及 MATLAB 专业人员的参考手册。

图书在版编目（CIP）数据

MATLAB 完全学习手册 / 赵国生等主编. —北京：清华大学出版社，2015
（完全学习手册）
ISBN 978-7-302-36806-9

Ⅰ. ①M…　Ⅱ. ①赵…　Ⅲ. ①Matlab 软件–手册　Ⅳ. ①TP317-62

中国版本图书馆 CIP 数据核字（2014）第 124344 号

责任编辑： 袁金敏
封面设计：
责任校对： 徐俊伟
责任印制：

出版发行： 清华大学出版社
网　　址： http://www.tup.com.cn, http://www.wqbook.com
地　　址： 北京清华大学学研大厦 A 座　　**邮　　编：** 100084
社 总 机： 010-62770175　　**邮　　购：** 010-62786544
投稿与读者服务： 010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈： 010-62772015，zhiliang@tup.tsinghua.edu.cn
印 刷 者：
装 订 者：
经　　销： 全国新华书店
开　　本： 185mm×260mm　　**印　张：** 37　　**字　　数：** 927 千字
版　　次： 2015 年 1 月第 1 版　　**印　　次：** 2015 年 1 月第 1 次印刷
附光盘
印　　数： ～
定　　价： 元

产品编号：056307-01

前　　言

基本内容

MATLAB 是 Matrix&Laboratory 两个词的组合，意为矩阵工厂（矩阵实验室），是由美国 MathWorks 公司出品的商业数学软件，用于算法开发、数据可视化、数据分析及数值计算的高级计算语言和交互式环境。它将数值分析、矩阵计算、科学数据可视化及非线性动态系统的建模和仿真等诸多强大功能集成在一个易于使用的视窗环境中，为科学研究、工程设计及必须进行有效数值计算的众多科学领域提供了一种全面的解决方案，具有编程效率高、用户使用方便、扩充能力强和移植性好等特点，代表了当今国际科学计算软件的先进水平。经过 MathWorks 公司的不断完善，目前 MATLAB 已经发展成为国际上最优秀的高性能科学与工程计算软件之一。

本书针对入门读者的学习特点，结合作者多年使用 MATLAB 的教学和实践经验，由浅入深、图文并茂，详细介绍了数值计算、符号运算、图形图像和 Simulink 仿真等方面的内容。在讲解过程中配以大量实例，使读者循序渐进地熟悉软件、学习软件、掌握软件。每章都是从基础知识开始介绍，然后是实例分析，最后是习题练习，使理论与实践紧密结合，具体分为 11 章，各章主要内容如下。

第 1 章介绍了 MATLAB 的历史发展，主要功能及熟悉 MATLAB 的操作环境。

第 2 章介绍了 MATLAB 的数据类型及其操作函数。学习了数组、矩阵、多项式的创建方法及关系和逻辑及其运算方法。

第 3 章介绍了符号计算、符号表达式、运算精度、符号矩阵的计算和符号函数等内容。

第 4 章介绍了图像处理与图像分析的相关内容，包括二维基本绘图、三维基本绘图和图形处理实用技术等基本知识、特征操作及编辑特征。

第 5 章介绍了 M 文件涉及的脚本、函数和程序调试等基础知识。

第 6 章介绍了 Simuink 的常用模块集、子系统及其封装、模型仿真和模型调试等内容。

第 7 章介绍了图形用户界面的组成，涉及组成图形用户界面的窗口、菜单、按钮及文字说明等各种对象。

第 8 章介绍了 MATLAB 科学计算问题的求解方法，内容涉及线性方程、非线性方程及常微分方程的求解、数据插值、数值积分和优化等方面。

第 9 章介绍了 S-函数，使用各类模板生成 S-函数，重点介绍了 C MEX 文件型 S-函数的编写方法。

第 10 章介绍了 MATLAB 工具箱的使用方法，重点介绍了神经网络工具箱和模糊逻辑工具箱两个应用较广的领域性工具箱。

第 11 章介绍了 MATLAB 对磁盘文件的访问和 MATLAB 平台与其他平台间的外部

接口。

主要特点

本书作者都是长期使用 MATLAB 进行教学和科研工作的教师和工程师，有着丰富的教学和编著经验。在内容编排上，按照读者学习的一般规律，结合大量实例讲解操作步骤，能帮助读者快速、真正地掌握 MATLAB 软件的使用。

本书具有以下鲜明的特点。

- ❑ 从零开始，轻松入门；
- ❑ 图解案例，清晰直观；
- ❑ 图文并茂，操作简单；
- ❑ 实例引导，专业经典；
- ❑ 学以致用，注重实践。

读者对象

- ❑ 学习 MATLAB 设计的初级读者。
- ❑ 具有一定 MATLAB 基础知识、希望进一步深入掌握 MATLAB 技术的中级读者。
- ❑ 大、中专院校理工科相关专业的学生。
- ❑ 从事科学计算、MATLAB 接口编程及图形处理的工程技术人员。

本书既可以作为理工科相关院校的教材，也可以作为读者自学的教程，同时也非常适合作为广大专业科研人员的参考手册。

本书由哈尔滨师范大学赵国生老师主编，宋一兵主审。此外，黑龙江工程学院于翔老师和哈尔滨理工大学王健老师也参与了本书的编写。赵国生老师主要编写第 1～7 章的内容，于翔老师编写第 8～9 章内容，王健老师编写第 10～11 章内容。其他参与本书编写与审较的人员有宋一兵、郭方方、刘海龙、苏岩、孙涛、那锐、李振兴、管殿柱、赵景波、王献红、李文秋等老师。在此一并表示感谢。

本书得到以下项目的支持：国家自然科学青年基金项目“可生存系统的自主认知模式研究”（61202458）、高等学校博士点专项基金项目“任务关键系统可信性增强的自律机理研究”（20112303120007）和中国博士后科学基金面上资助项目“认知网络系统的形式化建模与分析方法”（20090460882）。

感谢您选择了本书，希望我们的努力对您的工作和学习有所帮助，也希望您把对本书的意见和建议告诉我们。

零点工作室网站地址：www.zerobook.net
零点工作室联系信箱：gdz_zero@126.com

目　录

目录



目录

第 1 章　MATLAB 概述

MATLAB 是 MATrix LABoratory（矩阵实验室）的缩写，是由美国 MathWorks 公司于 20 世纪 80 年代初推出的一套以矩阵计算为基础、适合多学科、多种工作平台的功能强劲的大型软件。MATLAB 将科学计算、数据可视化、系统仿真和交互式程序设计功能集成在非常便于使用的环境中，具有编程效率高、用户使用方便、扩充能力强、移植性好等特点。经过 MathWorks 公司的不断完善，目前 MATLAB 已经发展成为国际上最优秀的高性能科学与工程计算软件之一。

通过对本章的学习，任何无基础的初学者都可以轻松地进入到 MATLAB 的殿堂，初步了解 MATLAB 的发展历史，掌握 MATLAB 的主要功能，熟悉 MATLAB 的操作环境，为后面的进一步学习打下坚实的基础。

1.1　MATLAB 简介

MATLAB 和 MATHEMATICA、MAPLE 并称为三大数学软件。它在数学类科技应用软件中的数值计算方面首屈一指。MATLAB 将数值分析、矩阵计算、科学数据可视化及非线性动态系统的建模和仿真等诸多强大功能集成在一个易于使用的视窗环境中，为科学研究、工程设计及必须进行有效数值计算的众多科学领域提供了一种全面的解决方案，并在很大程度上摆脱了传统非交互式程序设计语言（如 C、Fortran）的编辑模式，代表了当今国际科学计算软件的先进水平。

MATLAB 软件提供了大量的工具箱，可用于工程计算、控制设计、信号处理与通信、图像处理、信号检测、金融建模设计与分析等领域，解决这些应用领域内特定类型的问题。MATLAB 的基本数据单位是矩阵，符合科技人员对数学表达式的书写格式，总之，MATLAB 具有易学、适用范围广、功能强、开放性强、网络资源丰富等特点。

1．界面友好，容易使用

MATLAB 软件中有很多工具，这些工具基本都采用图形用户界面。MATLAB 的用户界面非常接近 Windows 的标准界面，操作简单，界面友好。最新的 MATLAB 版本提供了完整的联机查询和帮助系统，极大地方便了用户的使用。MATLAB 软件提供的 M 文件调试环境也非常简单，能够很好地报告出现的错误及原因。MATLAB 软件是采用 C 语言开发的，其流程控制语句和语法与 C 语言相近。如果初学者有 C 语言基础，会很轻松地掌握 MATLAB 编程和开发。MATLAB 编程语言非常符合科技人员对数学表达式的书写格式，便于非计算机专业人员使用。MATLAB 语言可移植性好、可拓展性强，已经广泛应用于科学研究及工程计算各个领域。

2．强大的科学计算和数据处理能力

MATLAB 软件的内部函数库提供了丰富的函数，方便实现用户所需的各种科学计算和数据处理功能，这些函数所采用的算法包含了科研和工程计算中的最新研究成果，并经过各种优化和容错处理。这些内部函数经过了无数次的检验，稳定性好，出错的可能性非常小。利用 MATLAB 软件进行科学计算和数据处理，是站在巨人的肩膀上，可以节省用户大量编程时间。用户可以将主要精力放到更具有创造性的工作上，把繁琐的底层工作交给 MATLAB 软件的内部函数。

3．强大的图形处理功能

MATLAB 软件具有非常强大的数据可视化功能，方便绘制各种复杂的二维图形、三维图形和多维图形。MATLAB 具有强大的图形处理功能，自带很多的绘图函数，方便为图形添加标注、标题和坐标轴等。MATLAB 2010a 对于三维图形，还可以设置视角、色彩控制及光照效果等。此外，MATLAB 软件还可以创建三维动画效果及隐函数绘图等，用于科学计算和工程绘图。

4．应用广泛的专业领域工具箱

在 MATLAB 软件对许多专门的领域都开发了功能强大的工具箱，在 MATLAB 2010a 软件中共有 40 多个工具箱，这些工具箱都是由特定领域的专家开发，用户可以直接使用工具箱学习、应用和评估不同的方法而不需要自己编写代码。MATLAB 工具箱中的函数源代码都是可读和可修改的，用户可通过对源程序的修改或加入自己编写的程序构造新的专用工具箱。在本章的 1.8 节列出了 MATLAB 软件的常用工具箱，本书将详细介绍这些工具箱，例如，符号计算工具箱、信号处理工具箱、图像处理工具箱、小波分析工具箱和神经网络工具箱等。

5．实用的程序接口

MATLAB 软件是一个开放的平台。通过 MATLAB 软件的外部程序接口，用户可以利用 MATLAB 同其他的开发语言或软件进行交互，发挥各自优势，从而提高工作效率。利用 MATLAB 软件的编译器可以将 M 文件转换为可执行文件或动态链接库，可以独立于 MATLAB 软件运行。在 MATLAB 软件中，还可以调用 C/C++语言、Fortran 语言和 Java 语言等编写的程序。此外，MATLAB 软件还可以和办公软件（如 Word 和 Excel 软件等）进行很好的交互。

1.2 MATLAB 的安装、退出及卸载

MATLAB 的安装非常简单，将 MATLAB 安装光盘插入到光驱，然后直接运行 setup.exe 进行安装。下面详细介绍 MATLAB 的安装、退出和卸载过程。

1.2.1 MATLAB 的安装

本书以 MATLAB 2010a 为例，介绍 MATLAB 的安装过程。

（1）进入 MATLAB 的安装目录，单击 setup.exe 文件，显示准备安装，然后开始安装，

并弹出如图 1-1 所示的对话框。两个单选按钮中，前者为应用 Internet 进行安装，后者为不用 Internet 进行安装，二者没有太大区别，通常选择后者。本书选择不用 Internet 进行安装。单击“Next”按钮，进入下一步。

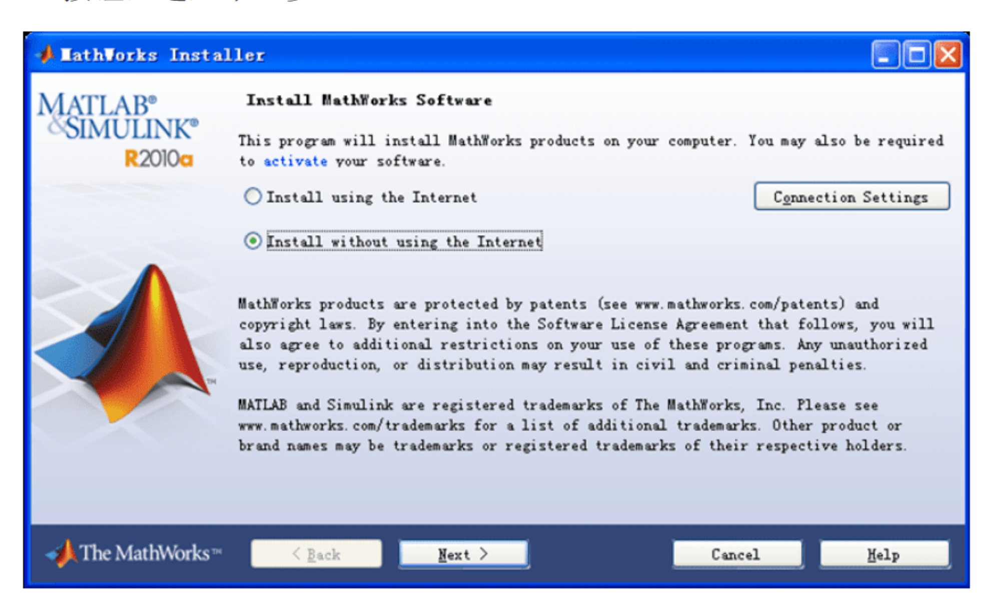


图 1-1 “MATLAB 2010a 安装”对话框

（2）单击 Next > 按钮后，显示“软件许可协议”对话框，如图 1-2 所示。选择“Yes”单选按钮接受软件许可协议，然后单击 Next > 进行下一步的安装。

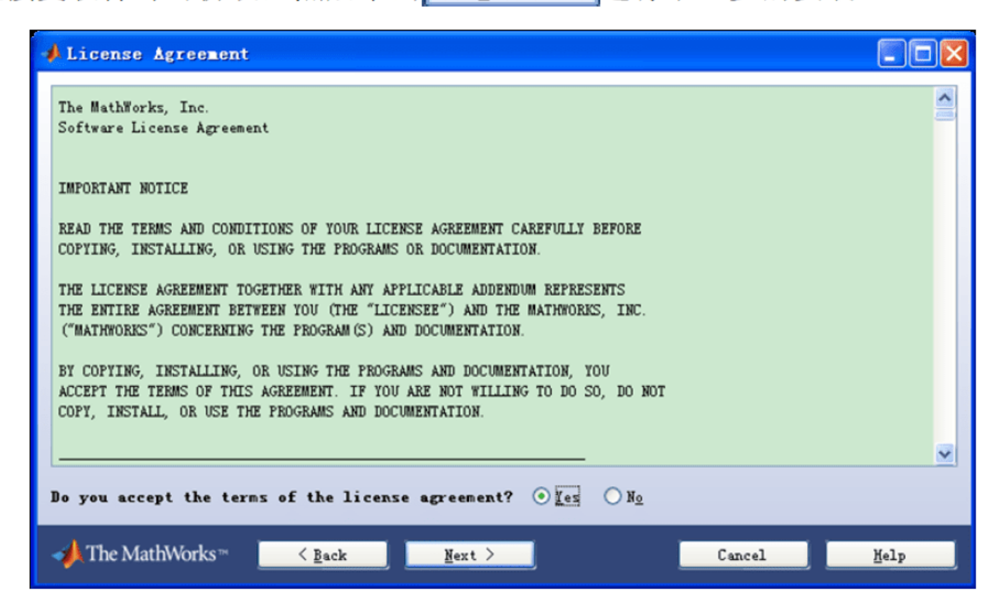


图 1-2 “软件许可协议”对话框

此时进入“输入安装序列号”对话框，如图 1-3 所示。在其中填入 MATLAB 2010a 的序列号，单击 Next > 按钮，进入下一步。

（3）进入的“安装类型”对话框中，有 Typical 和 Custom 两个选项，如图 1-4 所示。如果选择 Typical 选项，系统将会自动安装最常用的工具箱。如果选择 Custom 选项，用户可以根据自己的实际需要选择需要安装的工具箱。本书选择 Custom 选项，然后单击 Next > 按钮。

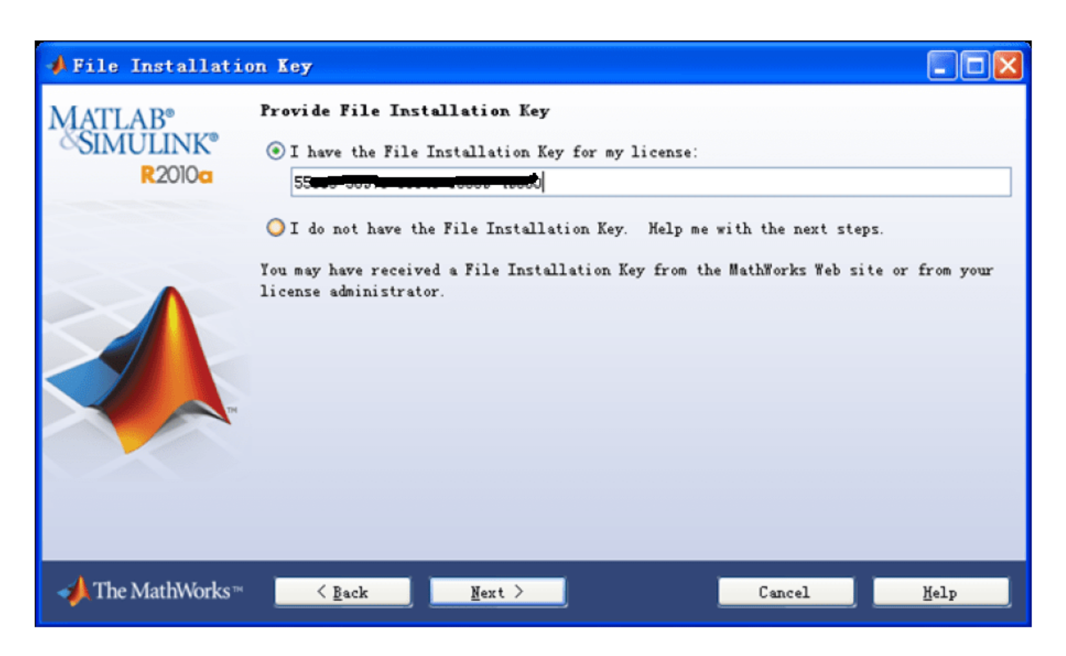


图 1-3 “输入序列号”对话框

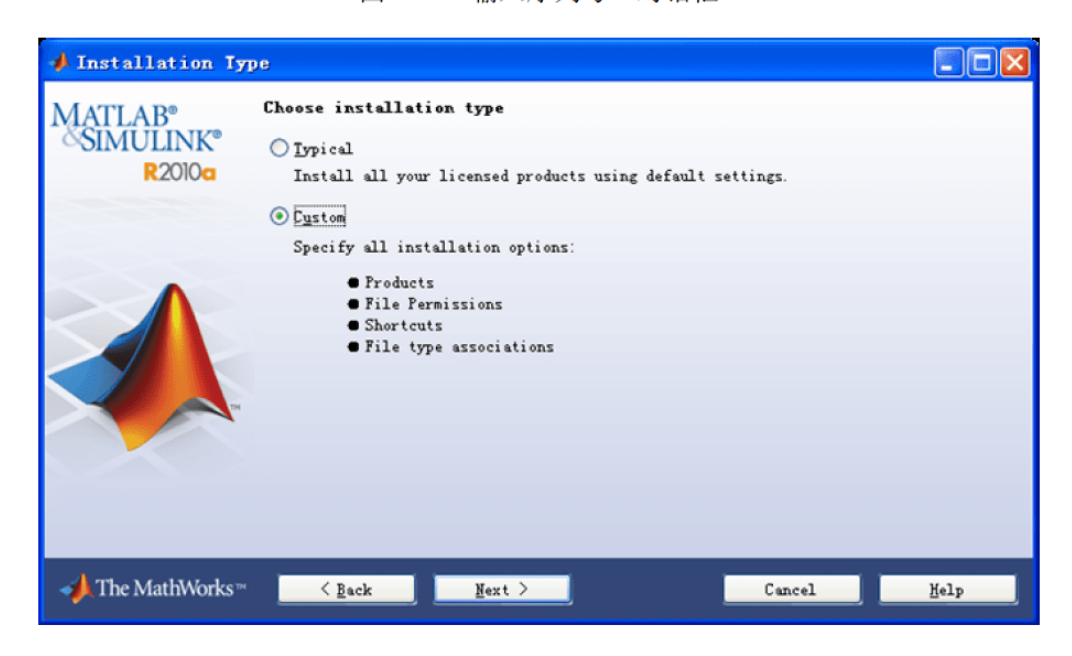


图 1-4 “安装类型”对话框

（4）出现“安装路径选择”对话框，如图 1-5 所示。用户可以单击 Browse 按钮选择需要安装的路径。系统默认安装路径为 C 盘的 C:\Program Files\MATLAB\R2010a。单击 Next > 按钮进入下一步。

（5）出现“工具箱选择”对话框，如图 1-6 所示。用户可以选择工具箱前面的复选框，来选择是否安装该工具箱，系统默认为所有的工具箱都为选中状态。如果所有的工具箱都安装，大概需要将近 6G 的空间。本书选择所有的工具箱，单击 Next > 按钮进入下一步。

进入“安装选项”对话框，如图 1-7 所示，在该对话框中可以设置是否在桌面和启动菜单添加快捷方式，以及和 MATLAB 相关的文件类型等，单击 Next > 按钮进入下一步。

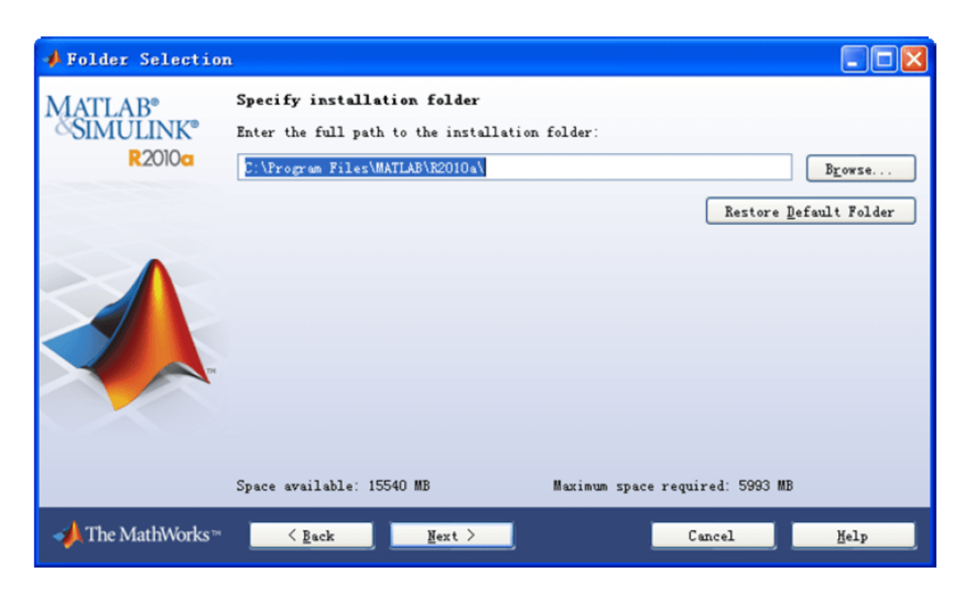


图 1-5 “安装路径选择”对话框

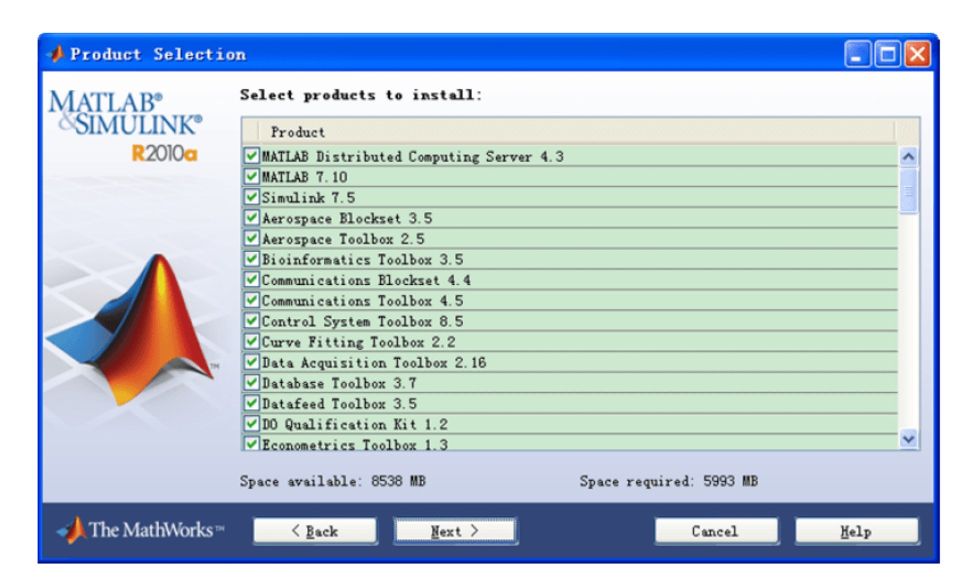


图 1-6 “工具箱选择”对话框

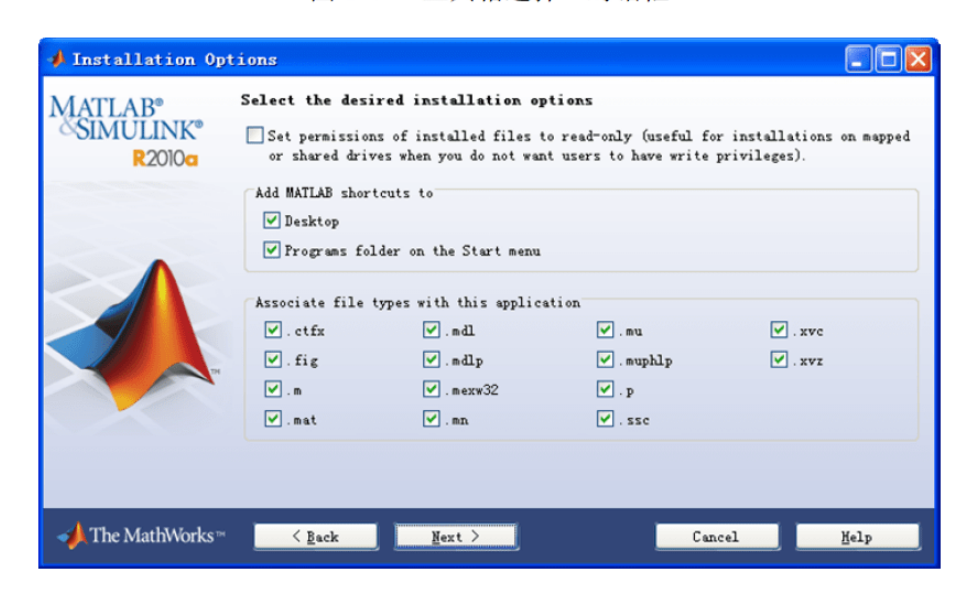


图 1-7 “安装选项”对话框

（6）进入“安装确认”对话框，如图 1-8 所示。本书选择的 MATLAB 安装路径为 E:\MATLAB，选择了所有的工具箱。如果用户的硬盘空间足够，建议完整安装所有的工具箱。之后单击 Install 按钮进行安装。

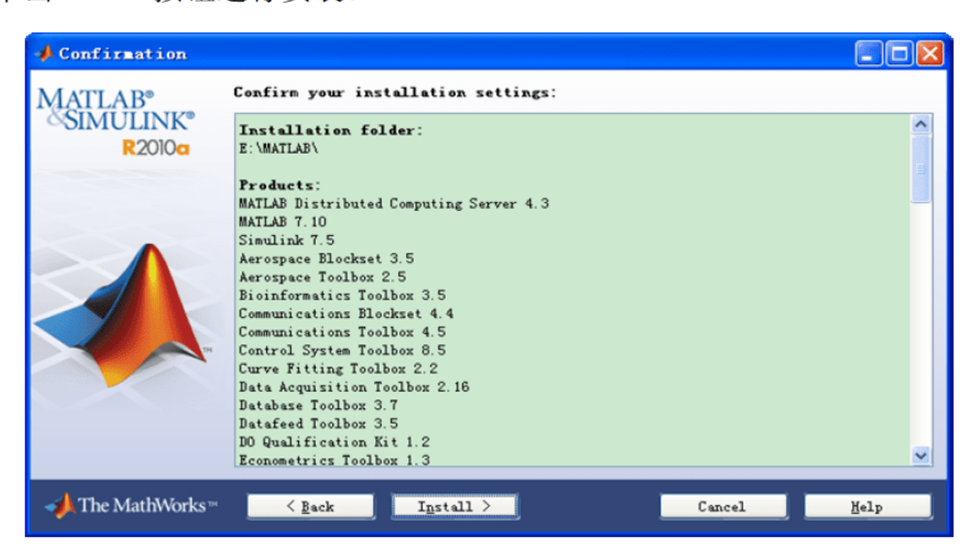


图 1-8 “安装确认”对话框

（7）开始 MATLAB 2010a 的正式安装，并显示安装进度，如图 1-9 所示。安装速度取决于计算机的硬件配置，以及选择的工具箱个数。

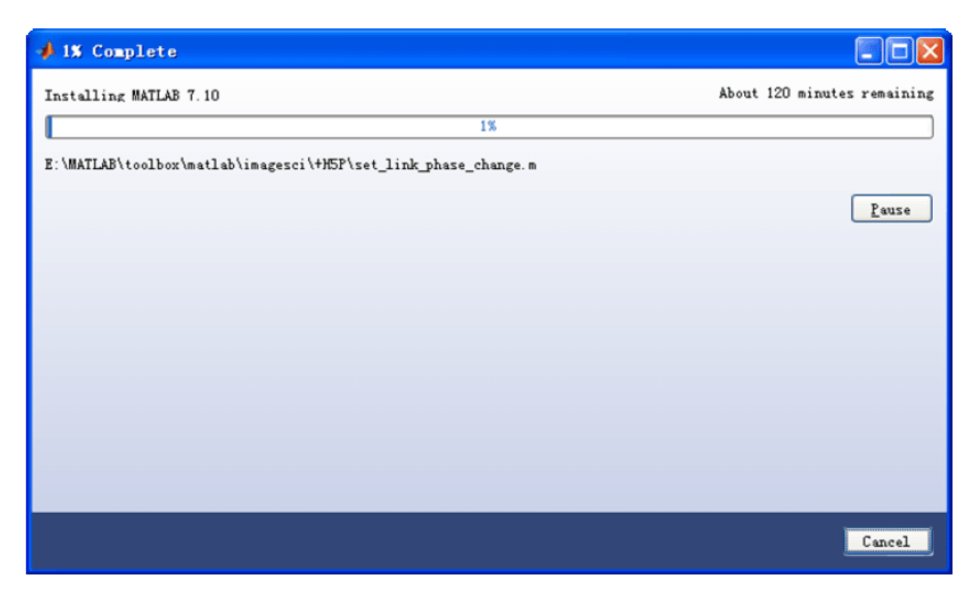


图 1-9 “安装进度”对话框

1.2.2 MATLAB 的启动和退出

MATLAB 2010a 安装结束后，可以通过单击“开始”菜单中的 MATLAB 来启动 MATLAB 系统，也可以在 MATLAB 的安装目录下找到 MATLAB.exe，然后单击运行。此外，用户可以在桌面建立 MATLAB 快捷菜单，通过双击快捷方式图标，启动 MATLAB 系统。

MATLAB 默认的启动目录是 C:\Documents and Settings\Administrator\My Documents\

MATLAB，可以进行修改。右击桌面上的 MATLAB R2010a 快捷图标，在弹出的快捷菜单中选择“属性”命令，弹出快捷菜单的属性设置窗口，如图 1-10 所示。设置 MATLAB 的初始目录为 D:\MATLAB2011\Program\chap1。

有以下 3 种方法可以退出 MATLAB 软件。

（1）在 MATLAB 的主窗口中选择【File】/【Exit MATLAB】命令或按快捷键 Ctrl+Q。

（2）在 MATLAB 的命令行窗口中输入 exit 或 quit。

（3）单击 MATLAB 主窗口右上角的关闭按钮进行关闭。

1.2.3　MATLAB 的卸载

用户如果想卸载 MATLAB 软件，可以通过在 Windows 控制面板中的添加或删除程序来卸载 MATLAB 软件，如图 1-11 所示。

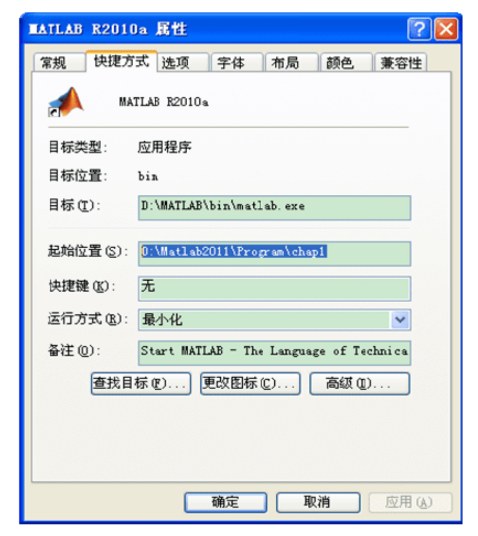


图 1-10　设置初始目录

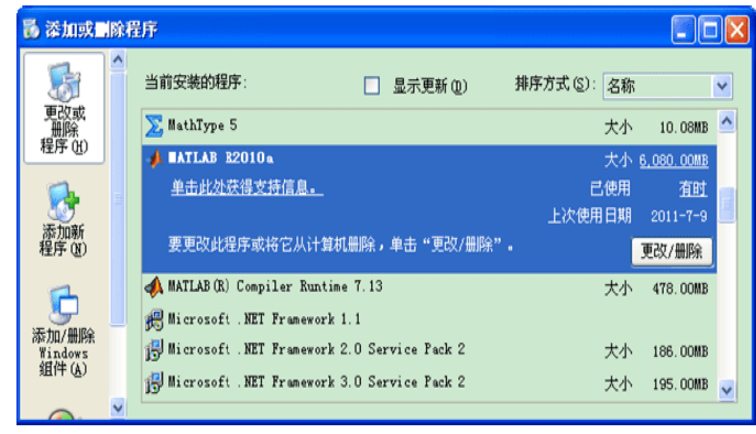


图 1-11　控制面板的“添加/删除程序”对话框

在图 1-11 中，单击“更改/删除”按钮，弹出对话框，如图 1-12 所示，用户可以在其中选择要卸载的程序或工具箱，系统默认全部程序和工具箱都为选中状态。单击 Uninstall 按钮，可进行 MATLAB 的卸载。

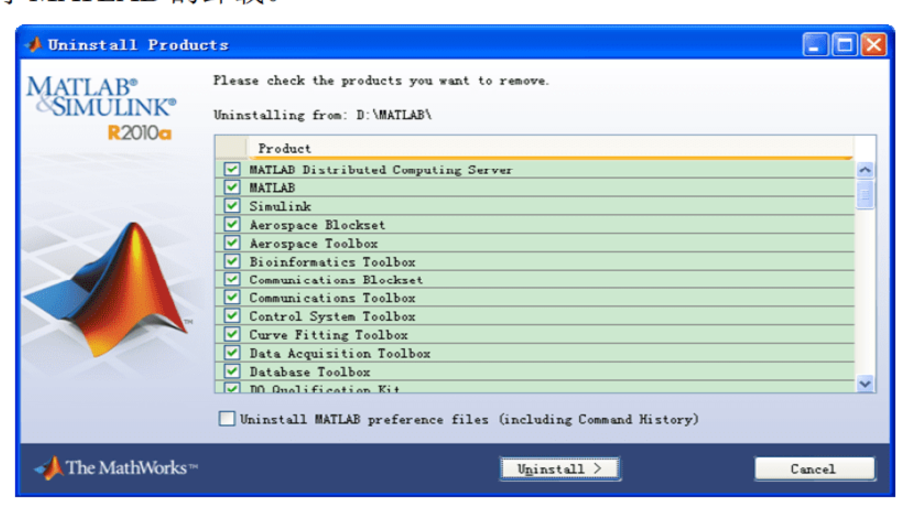


图 1-12　卸载 MATLAB 对话框

1.3 MATLAB 的目录结构

用户成功将 MATLAB 安装到 D:\MATLAB 目录后，该文件夹中的文件如图 1-13 所示，用户可以查阅各文件夹的内容。

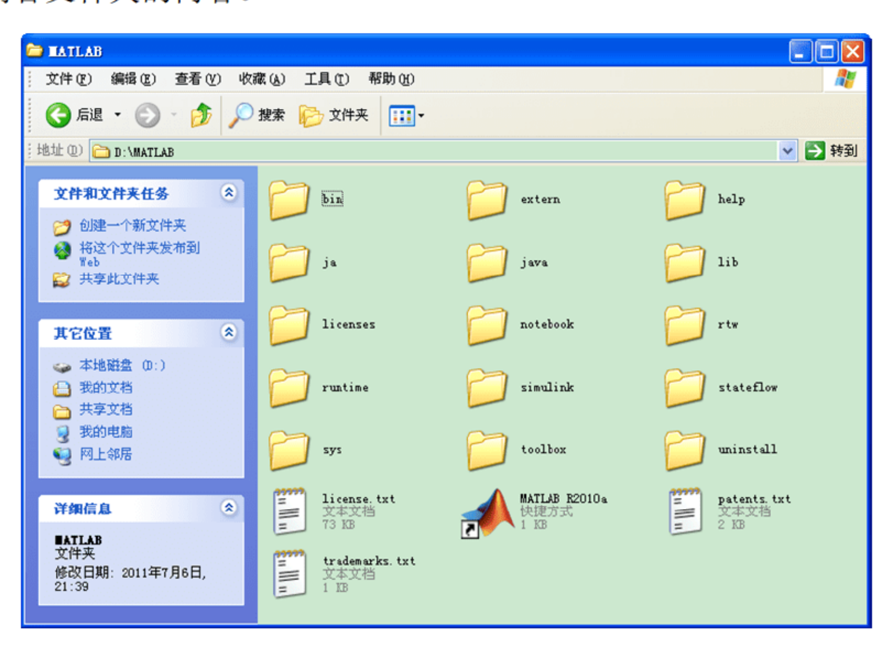


图 1-13　MATLAB 安装后的目录

在图 1-13 中，各文件夹的内容介绍如表 1-1 所示。可以单击快捷方式 MATLAB R2010a 来启动 MATLAB 软件。

表 1-1　MATLAB 安装文件夹的目录结构

文　件　夹	说　　明
bin	MATLAB 的可执行文件
extern	MATLAB 的外部程序接口
help	MATLAB 的帮助系统
ja	MATLAB 的国际化文件
java	MATLAB 的 Java 支持程序
lib	几个库文件
license	MATLAB 软件的许可协议
notebook	MATLAB 和 Word 的接口文件
rtw	Real-TIme Workshop 软件包
runtime	运行时库
simulink	Simulink 软件包，用于系统的建模和仿真
stateflow	Stateflow 软件包，用于状态机的设计
sys	MATLAB 所需的工具和系统库
toolbox	MATLAB 的各种工具箱
uninstall	MATLAB 的卸载程序

1.4 MATLAB 的应用窗口

窗口是指某一应用程序的使用界面。在图形界面操作系统中，窗口是其最重要的组成部分之一。下面来认识 MATLAB 2010a 运行中一系列具体的应用窗口。

MATLAB 2010a 的工作界面如图 1-14 所示，包括菜单、工具栏、当前工作目录、命令行窗口、工作空间窗口和历史命令窗口。

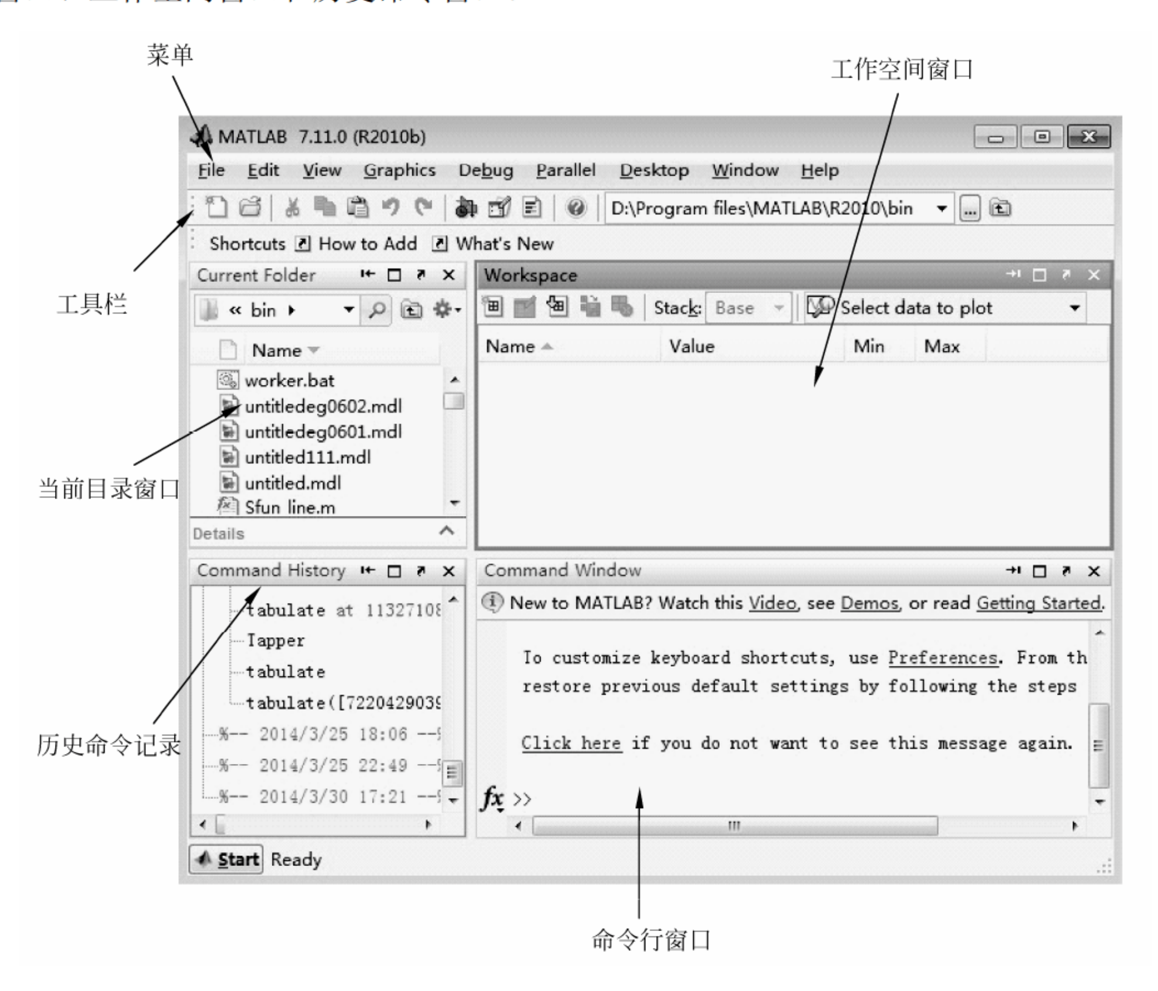


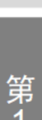
图 1-14 MATLAB 的工作界面

1.4.1 工具栏

表 1-2 所示简单列出了工具栏中各按钮图例及其功能。

表 1-2 工具栏中各按钮控件的图例及其功能

图　例	按钮控件的功能
	用 MATLAB 的 M 文件编辑调试器新建一个文件
	用 MATLAB 的 M 文件编辑调试器打开一个文件



续表

图　例	按钮控件的功能
	将选中的内容剪切到剪贴板上
	复制选中的内容
	将剪贴板上的内容粘贴到指定的位置
	撤销上一次的操作
	恢复上一次的操作
	打开 Simulink
	打开 GUIDE
	打开 Profiler
	打开 MATLAB 帮助系统

1.4.2　组件窗口

1．命令窗口（Command Window）

MATLAB 的命令窗口是用户使用 MATLAB 进行工作的窗口，同时也是实现 MATLAB 各种功能的主窗口，MATLAB 的各种操作命令都是由命令窗口开始的。用户可以直接在 MATLAB 命令窗口中输入 MATLAB 命令，实现其相应的功能，此命令窗口主要包括文本的编辑区域和菜单栏，如图 1-15 所示。

2．M 文件编辑/调试器窗口（Editor/Debugger）

M 文件编辑/调试器是用户在 MATLAB 中进行程序设计，实现函数功能的重要编辑器之一，其窗口界面如图 1-16 所示。

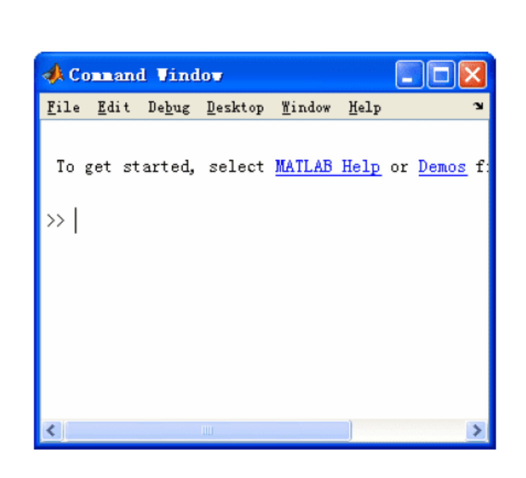


图 1-15　MATLAB 命令窗口

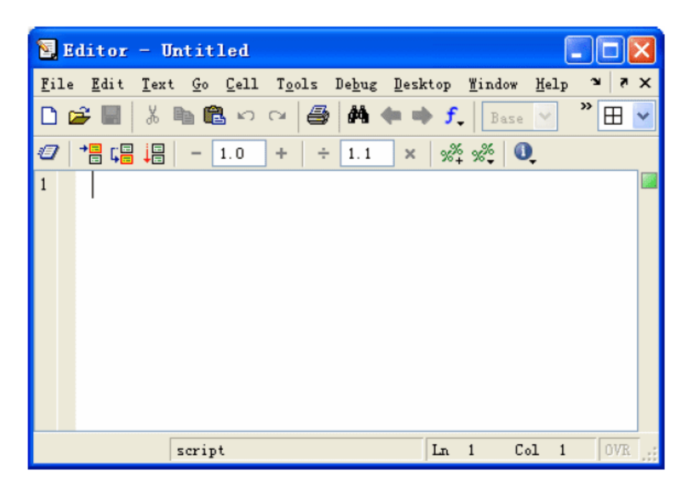


图 1-16　M 文件编辑/调试器窗口

下面只对 M 文件编辑/调试器的 Edit 菜单中的重要部分内容进行介绍。

- 选择【Edit】/【Find and Replace】命令可打开查找和替换对话框，如图 1-17 所示。
- 选择【Edit】/【Find Next】命令可以查找下一个符合条件的字符。
- 选择【Edit】/【Find Selection】命令可以查找与所选字符相匹配的字符。

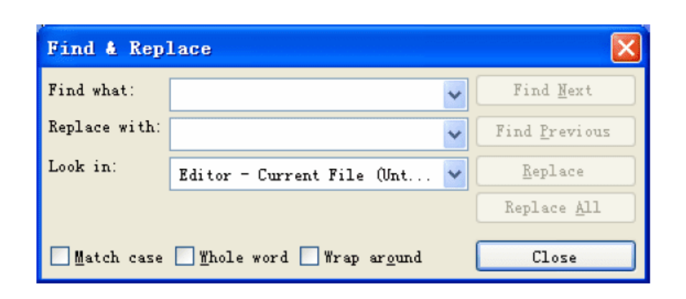


图 1-17　Find & Replace 对话框

M 文件编辑/调试器的工具栏如图 1-18 所示。



图 1-18　M 文件编辑/调试器的工具栏

下面只对此工具栏中特殊的按钮控件进行叙述，如表 1-3 所示。

表 1-3　工具栏中特殊的按钮控件

图　　例	按钮控件的功能
	相当于 Edit（编辑）菜单中的 Find Next 命令
	后退一步
	前进一步
	显示函数
	设置/取消指定行的断点
	清除所有 M 文件中的断点
	逐步执行程序
	进入子函数中逐步执行程序
	跳出子函数
	保存后继续执行调试程序
	退出调试状态
	M 文件全部显示
	M 文件左右显示
	M 文件上下显示
	M 文件浮动
	M 文件最大化
	HTML 格式显示
	计算数组
	计算前面所有数组
	计算整个文件
− 1.0	指针减小并计算数组

续表

图　　例	按钮控件的功能
1.0 +	指针增大并计算数组
÷ 1.1	指针被除并计算数组
1.1 ×	指针被乘并计算数组
	插入数组
	显示数组标题
	显示数组模式信息

3．图形窗口

MATLAB 的图形窗口是 MATLAB 绘图功能的基础，使用极其方便，其菜单和工具栏更是增添了交互处理的功能。

（1）图形窗口的菜单栏

与桌面平台的 File 菜单相近，只是增加了图形输出 Generate M-file、Export Setup、Print Preview 和 Print 命令。

❑ 选择【File】/【Export Setup】命令打开如图 1-19 所示的“Export Setup”对话框。

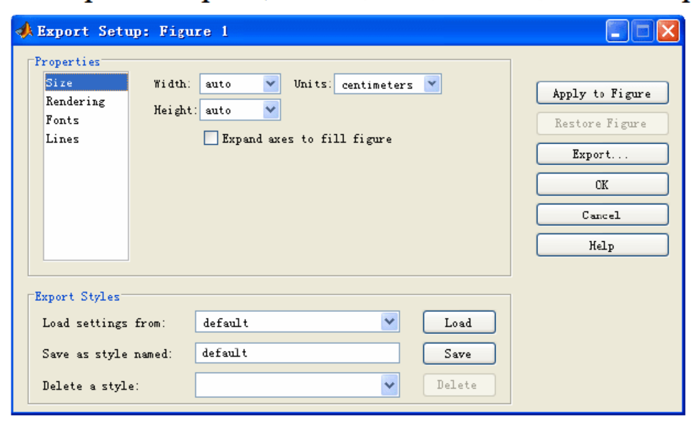


图 1-19 “Export Setup”对话框

❑ 选择【File】/【Page Setup】命令打开如图 1-20 所示的“Page Setup”对话框。

❑ 选择【File】/【Print Setup】命令打开如图 1-21 所示的“打印设置“对话框。

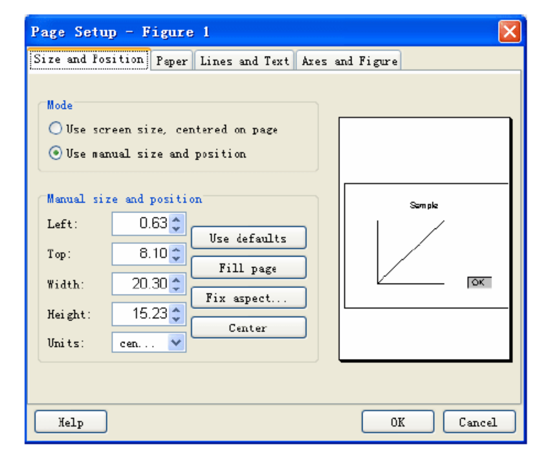


图 1-20 “Page Setup”对话框

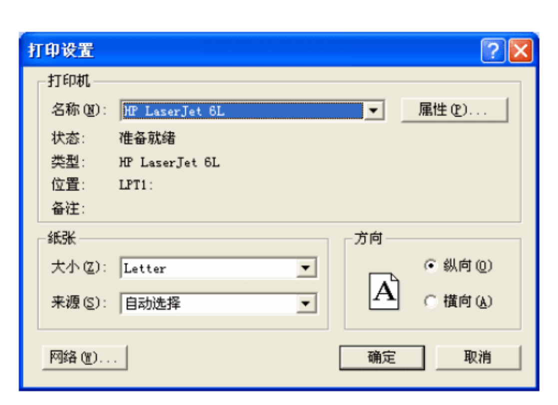


图 1-21 “打印设置”对话框

❑ 选择【File】/【Print Preview】命令打开如图 1-22 所示的“Print Preview”对话框。

另外，图形窗口的 Desktop（桌面）菜单、Window（窗口）菜单和 Help（帮助）菜单，与其他系统大致相同，也比较简单，可以对照学习，在此不再叙述。

（2）图形窗口的工具栏

图形窗口的工具栏位于菜单栏的下面，如图 1-23 所示。

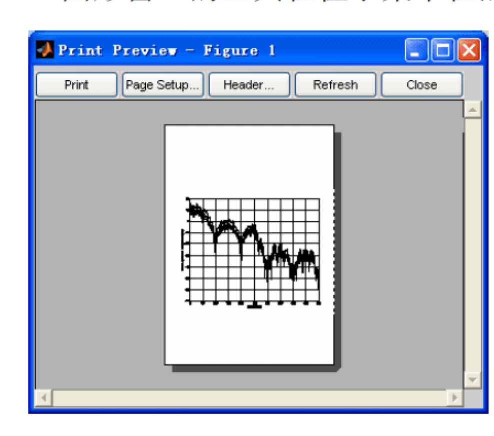


图 1-22 “Print Preview”对话框



图 1-23 图形窗口的工具栏

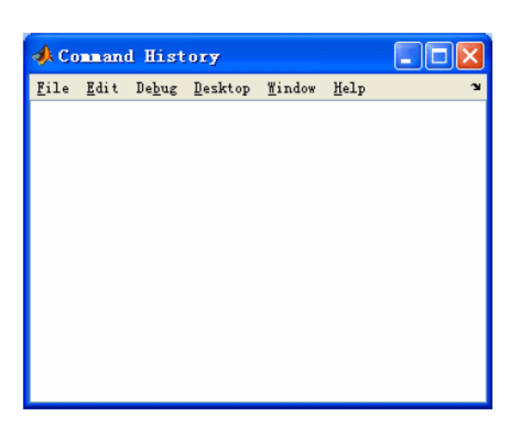
表 1-4 简单列出工具栏中各按钮控件的图例及功能。

表 1-4 工具栏各按钮控件的图例及功能

图 例	按钮控件的功能
	新建一个图形文件
	打开一个图形文件
	以.fig 的格式保存图形文件
	打印图形文件
	使图形窗口处于被编辑状态
	放大图形
	缩小图形
	拖动图形
	对图形进行三维手动旋转
	数据指针
	插入颜色工具栏
	插入图例
	隐藏绘图工具
	显示绘图工具

4. 历史命令（Command History）窗口

历史命令窗口界面如图 1-24 所示。在历史命令窗口单击鼠标右键，打开如图 1-25 所示的快捷菜单。



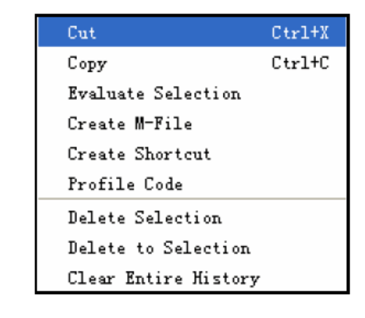
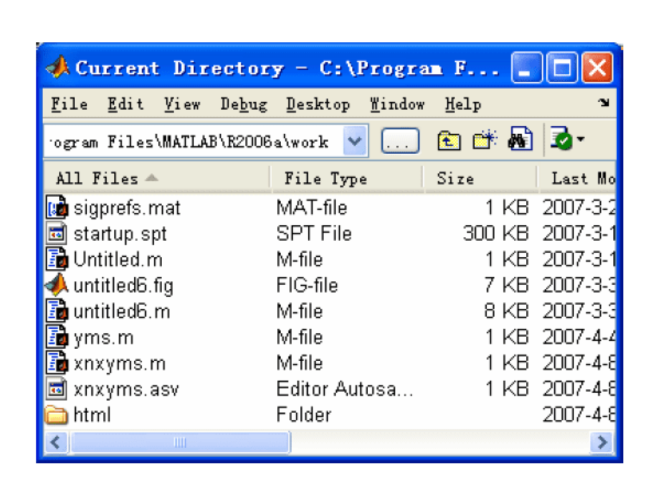


图 1-24　历史命令窗口　　　　图 1-25　历史命令窗口的快捷菜单

5．当前路径窗口（Current Directory）

当前路径窗口显示当前路径下的文件，如图 1-26 所示。在当前路径窗口中单击鼠标右键，打开如图 1-27 所示的快捷菜单。



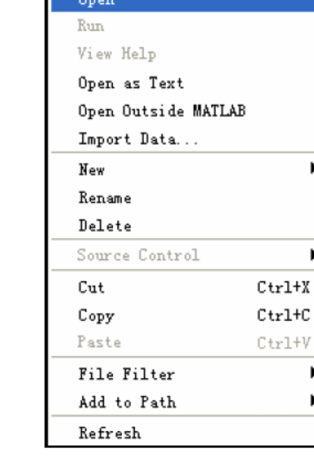


图 1-26　当前路径窗口　　　　图 1-27　当前路径窗口的快捷菜单

6．工作空间窗口（Workspace）

工作空间窗口就是显示目前保存在内存中的 MATLAB 数学结构、字节数、变量名及类型等的窗口，如图 1-28 所示。

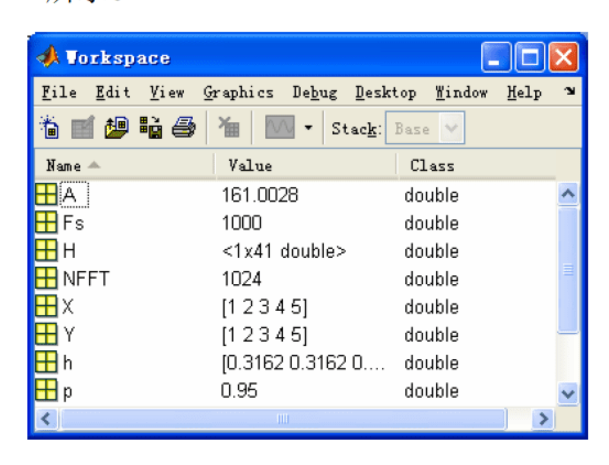


图 1-28　工作空间窗口

在工作空间窗口单击鼠标右键，打开如图 1-29 所示的快捷菜单。工作空间窗口快捷菜单中的 Open Selection 命令用于打开所选数据变量的数组编辑器，如图 1-30 所示。用户可以直接在数组编辑器对话框内修改数据的结构、数据和属性。

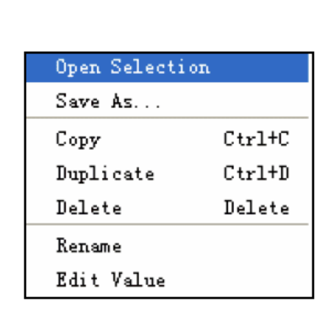


图 1-29　工作空间窗口的快捷菜单

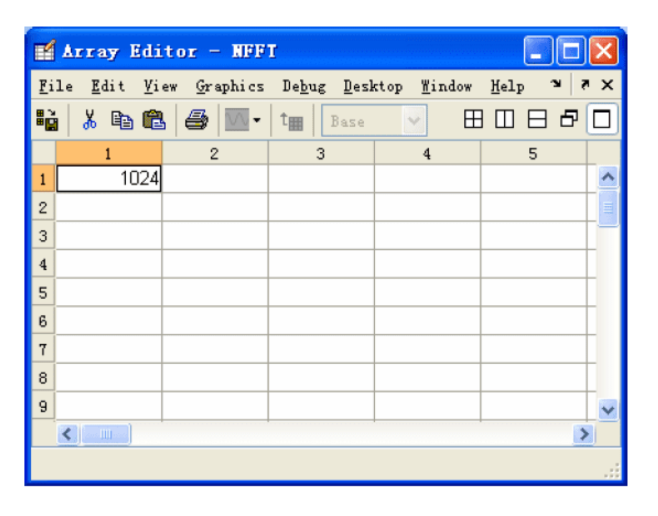


图 1-30　数组编辑器

工作空间浏览器窗口的工具栏如图 1-31 所示。

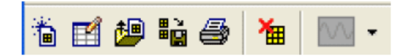


图 1-31　工作空间浏览器窗口的工具栏

其中，各按钮控件的功能如表 1-5 所示。

表 1-5　按钮控件的功能

图　　例	按钮控件的功能
	向工作空间添加新变量
	打开数组编辑器
	向工作空间载入数据
	保存工作空间的变量
	打印工作空间的变量
	删除工作空间的变量
	绘制工作空间的变量图

1.5　MATLAB 的通用命令

在 MATLAB 中，有很多命令经常用到，需要熟练掌握，例如，在命令行窗口输入命令：clc，清除命令行窗口中所显示的内容。MATLAB 的常用命令，如表 1-6 所示。

表 1-6　MATLAB 的常用命令

命　　令	说　　明	命　　令	说　　明
cd	改变当前目录	!	调用 DOS 命令
dir 或 ls	列出当前文件夹下的文件	edit	打开 M 文件编辑器
clc	清除命令行窗口的内容	mkdir	创建目录
type	显示文件内容	pwd	显示当前工作目录
clear	清除工作空间中的变量	what	显示当前目录下的 M 文件、MAT 和 MEX 文件
disp	显示文字内容	which	函数或文件的位置
exit 或 quit	关闭 MATLAB	help	获取函数的帮助信息
save	保存变量到磁盘	pack	收集内存碎片
load	从磁盘调入数据变量	path 或 genpath	显示搜索路径
who	列出工作空间中的变量名	clf	清除图形窗口的内容
whos	显示变量的详细信息	delete	删除文件

MATLAB 中的一些标点符号有特殊的含义，例如，利用百分号%进行程序的注释，利用…进行程序的续行。MATLAB 中常用的标点符号，如表 1-7 所示。

表 1-7　MATLAB 语言的标点符号

标 点 符 号	说　　明	标 点 符 号	说　　明
:	冒号，具有多种应用	.	小数点或对象的域访问
;	分号，区分矩阵的行或取消运行结果的显示	..	父目录
,	逗号，区分矩阵的列	…	续行符号
( )	括号，指定运算的顺序	!	感叹号，执行 DOS 命令
[ ]	方括号，定义矩阵	=	等号，用来赋值
{}	大括号，构造单元数组	'	单引号，定义字符串
@	创建函数句柄	%	百分号，程序的注释

在 MATLAB 中，键盘按键便于进行程序的编辑，有时可以起到事半功倍的效果，常用的键盘按键及其作用如表 1-8 所示。

表 1-8　常用的键盘按键

键 盘 按 键	说　　明	键 盘 按 键	说　　明
↑	调出前一个命令	→	光标向右移动一个字符
↓	调出后一个命令	Ctrl +←	光标向左移动一个单词
←	光标向左移动一个字符	Ctrl+→	光标向右移动一个单词
Home	光标移动到行首	Del	清除光标后的字符
End	光标移动到行尾	Backspace	清除光标前的字符
Esc	清除当前行	Ctrl +C	中断正在执行的程序

1.6　MATLAB 的帮助系统

MATLAB 提供了非常完善的帮助系统。用户可以通过查询帮助系统获取函数的调用情

况和需要的信息。对于任何 MATLAB 的使用者，必须学会使用 MATLAB 的帮助系统，因为没有人能够清楚地记住上万个不同函数的调用情况，所以 MATLAB 的帮助系统是学习 MATLAB 编程和开发最好的教科书，讲解清晰、易懂。下面对 MATLAB 的帮助系统进行介绍。

1.6.1 命令行窗口查询帮助

在 MATLAB 中，可以在命令行窗口中通过帮助命令来查询帮助信息，最常用的帮助命令是 help。常用的帮助命令如表 1-9 所示。

表 1-9 常用的帮助命令

命 令	说 明
help	在命令行窗口进行查询
which	获取函数或文件的路径
lookfor	查询指定关键字相关的 M 文件
helpwin	在浏览器中打开帮助窗口，可以带参数
helpdesk	在浏览器中打开帮助窗口，显示帮助的首页
doc	在帮助窗口中显示函数查询的结果
demo	在帮助窗口显示例子程序

在 MATLAB 的命令行窗口输入：help，输出结果为：

```
>> help
HELP topics:
MATLAB\general              - General purpose commands.
MATLAB\ops                  - Operators and special characters.
MATLAB\lang                 - Programming language constructs.
MATLAB\elmat                - Elementary matrices and matrix manipulation.
MATLAB\randfun              - Random matrices and random streams.
MATLAB\elfun                - Elementary math functions.
MATLAB\specfun              - Specialized math functions.
.................................................................
build\xpcblocks             - xPC Target -- Blocks
xpc\xpcdemos                - xPC Target -- demos and sample script files.
kernel\embedded             - xPC Target Embedded Option
```

1.6.2 MATLAB 联机帮助系统

用户可以选择 MATLAB 主界面的【Help】/【Product Help】命令，或在命令行窗口输入 helpdesk 或 doc 命令后，在浏览器中打开 MATLAB 的帮助系统，如图 1-32 所示。MATLAB 的帮助系统和以前版本的帮助系统有很大差别。

在 MATLAB 的命令行窗口输入 doc std，或在图 1-33 的查询窗口中输入 std，可以查询函数 std()的帮助信息，如图 1-33 所示。在左侧的 Search Results 选项中，列出了所有函数 std()的重载函数，用户可以用鼠标进行选择，并查看该函数的详细情况。

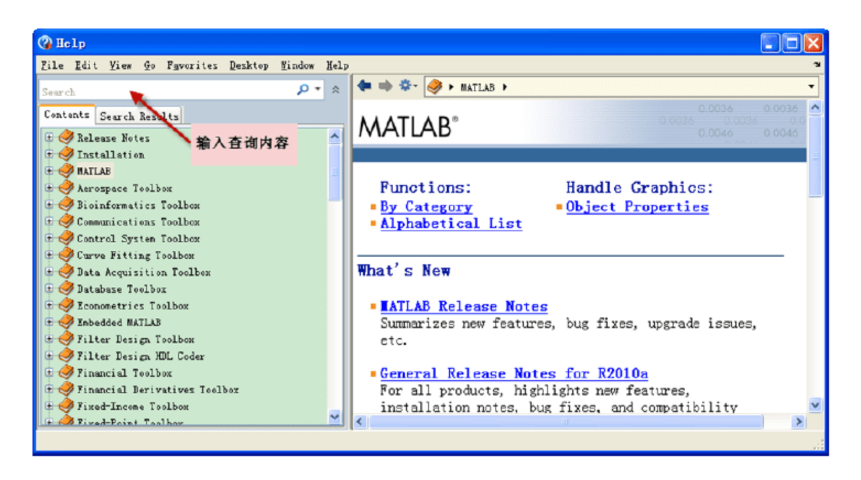


图 1-32　MATLAB 的查询界面

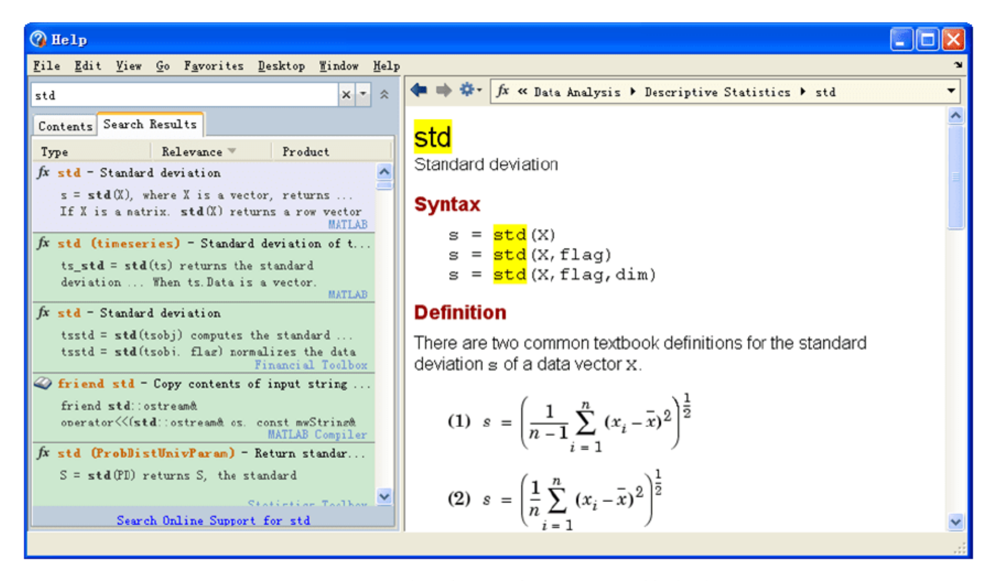


图 1-33　利用帮助系统进行函数查询

如果用户在命令行窗口输入 helpwin 命令后，MATLAB 的查询界面如图 1-34 所示。在图 1-34 中，MATLAB 的命令或函数按照列表进行了分类。例如，单击 MATLAB\general 后，将获得 MATLAB 系统的通用命令。如果在命令行窗口输入 help general，将会在命令行窗口显示 MATLAB 系统的通用命令。

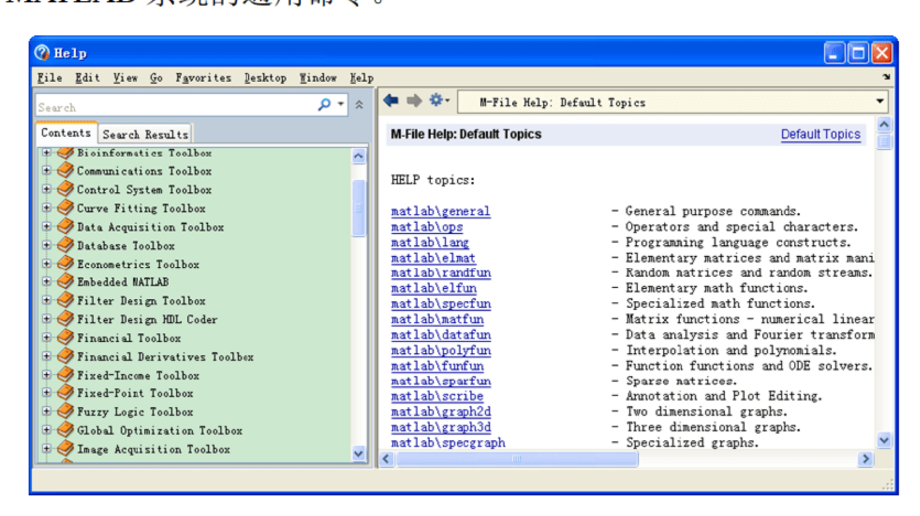


图 1-34　输入命令 helpwin 后的查询界面

1.7　本章小结

本章着重介绍了 MATLAB 的基础知识。首先介绍了 MATLAB 语言本身的历史、安装、启动和卸载过程。接下来为了使用户能尽快熟悉 MATLAB 的操作环境，对 MATLAB 重要的窗口界面进行了介绍。最后，又对 MATLAB 语言的联机帮助系统等进行了介绍。对用户来说，本章是全书学习的基础，只有掌握好本章的知识，才能更好地学习后面的内容。

1.8　习题

（1）与其他计算机语言相比较，MATLAB 语言突出的特点是什么？

（2）MATLAB 系统由哪些部分组成？

（3）MATLAB 操作桌面有几个窗口？如何使某个窗口脱离桌面成为独立窗口？又如何将脱离出去的窗口重新放置到桌面上？

（4）如何启动 M 文件编辑/调试器？

（5）如何设置当前目录和搜索路径，在当前目录上的文件和在搜索路径上的文件有什么区别？

（6）在 MATLAB 中有几种获得帮助的途径？



第 2 章　MATLAB 数值计算

本章介绍 MATLAB 的几种重要数据类型及其操作方法。同时，介绍 MATLAB 的几种重要的数值计算方法：矩阵、数组、多项式。

对于那些熟悉其他高级语言（如 FORTRAN，Pascal，C++）的读者来说，通过本章 MATLAB 卓越的数组处理能力、浩瀚而灵活的 M 函数指令、丰富而友善的图形显示指令将能体验到解题视野的豁然开朗，感受到摆脱烦琐编程后的眉眼舒展。

对于那些经过大学基本数学教程的读者来说，通过本章 MATLAB 精良完善的计算指令，自然易读的程序感悟“教程”数学的基础地位和局限性，看到从“理想化”简单算例通向科学研究和工程设计实际问题的一条途径。

从总体上讲，本章各节之间没有依从关系，即读者没有必要从头到尾系统阅读本章内容。完全可以根据需要阅读有关节次。除特别说明外，每节中的例题指令是独立完整的，因此读者可以在自己机器上实践。

2.1　数据类型

MATLAB 中定义了 15 种数据类型，基本数据类型是双精度数据类型和字符类型，如图 2-1 所示。MATLAB 的不同数据类型的变量或对象占用的内存空间不同，不同的数据类型的变量或对象也具有不同的操作函数。本节将讨论这些数据类型及其用法。

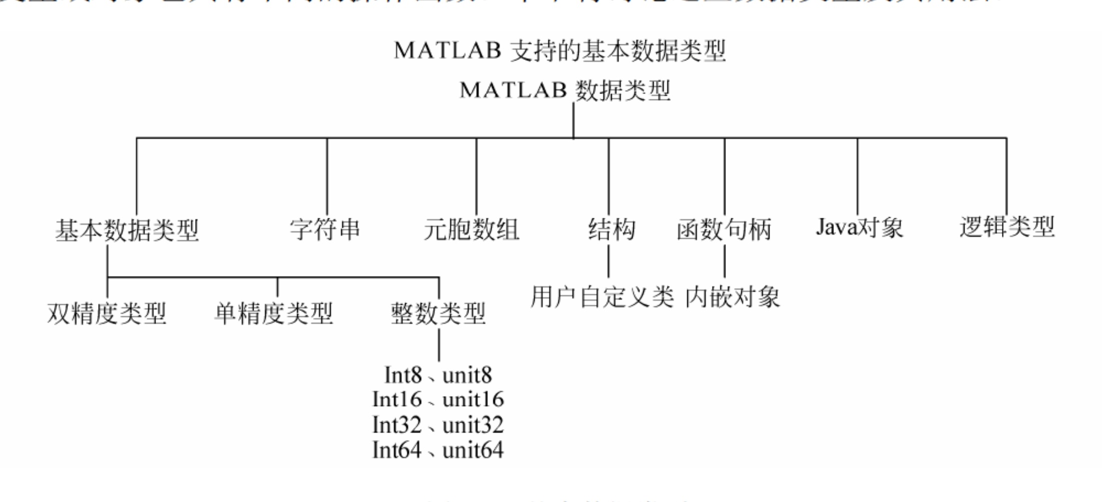


图 2-1　基本数据类型

2.1.1　字符串（String）

在 MATLAB 中可能会遇到对字符和字符串的操作。字符串能够显示在屏幕上，也可

以用来构成一些命令，这些命令在其他命令中用于求值或被执行。字符串在数据的可视化、应用程序的交互方面起到非常重要的作用。

一个字符串是存储在一个行向量中的文本，这个行向量中的每一个元素代表一个字符，每一个字符占用两个字节的内存。实际上，元素中存放的是字符的内部代码，也就是ASCII 码。当在屏幕上显示字符变量的值时，显示的是文本，而不是 ASCII 数字。由于字符串是以向量的形式来存储的，所以可以通过它的下标对字符串中的任何一个元素进行访问。字符矩阵也可以这样，但是它的每行字符数必须相同。

1. 字符串的创建方法

创建字符串时，只要将字符串的内容用单引号包括起来即可。

【例 2-1】 创建字符串。

```
>> a=127
a =
   127
>> class(a)
ans =
double
>> size(a)
ans =
     1     1
>> b='127'
b =
   127
>> class(b)
ans =
      char
>> size(b)
ans =
     1     3
```

若需要在字符串内容中包含单引号，则在键入字符串内容时，连续键入两个单引号即可。

使用 char 函数创建一些无法通过键盘输入的字符，该函数的作用是将输入的整数参数转变为相应的字符。

【例 2-2】 使用 char 函数创建一些无法通过键盘输入的字符。

```
>> S1=char('Good','Job.')
S1 =
     Good
     Job.
>> S2=char('祝','老师','教师节','','快乐')
S2 =
     祝
     老师
```

```
教师节
快乐
```

2. 字符串的基本操作

1）字符串元素索引

字符串实际上也是一种 MATLAB 的向量或数组，一般利用索引操作数组的方法都可以用来操作字符串。

2）字符串拼接

字符串可以利用“[]”运算符进行拼接。

若使用“,”作为不同字符串之间的间隔，相当于扩展字符串成为更长的字符串向量；若使用“;”作为不同字符串之间的间隔，则相当于扩展字符串成为二维或多维的数组，这时不同行上的字符串必须具有同样的长度。

3）字符串和数值的转换

使用 char 函数可以将数值转变为字符；使用 double 函数可以将字符转变成数值。

4）字符串操作函数

表 2-1　字符串操作函数

函　数	说　明
char	创建字符串，将数值转变成为字符串
double	将字符串转变成为 Unicode 数值
blanks	创建空白的字符串（由空格组成）
deblank	将字符串尾部的空格删除
ischar	判断变量是否是字符类型
strcat	水平组合字符串，构成更长的字符向量
strvcat	垂直组合字符串，构成字符串矩阵
strcmp	比较字符串，判断字符串是否一致
strncmp	比较字符串前 n 个字符，判断是否一致
strcmpi	比较字符串，比较时忽略字符的大小写
strncmpi	比较字符串前 n 个字符，比较时忽略字符的大小写
findstr	在较长的字符串中查寻较短的字符串出现的索引
strfind	在第一个字符串中查寻第二个字符串出现的索引
strjust	对齐排列字符串
strrep	替换字符串中的子串
strmatch	查询匹配的字符串
upper	将字符串的字符都转变成为大写字符
lower	将字符串的字符都转变成为小写字符

❑ **Blanks**　创建空白的字符串（由空格组成）。

【例 2-3】 使用 Blanks 创建空字符串。

```
>> a=blanks(4)
a =
```

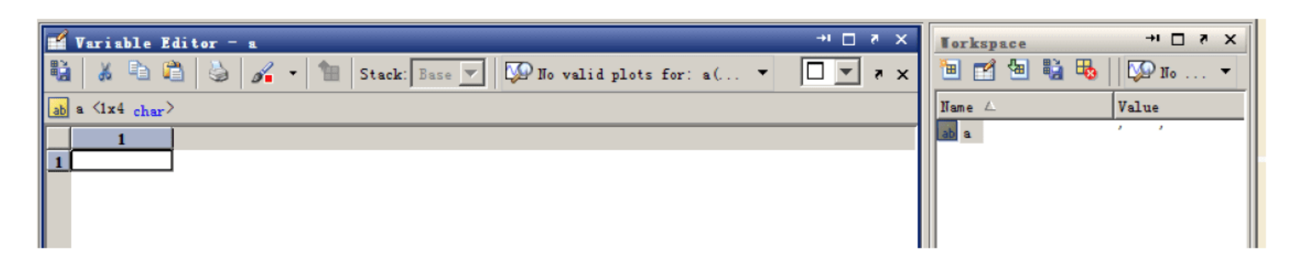


图 2-2　创建的空字符串

❑ **deblank**　将字符串尾部的空格删除。

【例 2-4】　使用 deblank 去掉字符串尾部空格。

```
>> a='Hello!   '
a =
Hello!
>> deblank(a)
ans =
Hello!
>> whos
  Name       Size                    Bytes  Class
  a          1x9                      18  char array
  ans        1x6                      12  char array
Grand total is 15 elements using 30 bytes
```

❑ **ischar**　判断变量是否是字符类型，变量为字符型，则结果为 1；变量不为字符型，则结果为 0。

【例 2-5】使用 ischar 判断变量是否为字符类型。

```
>> a='Hello!'
a =
Hello!
>> ischar(a)
ans =
    1
>> b=12;
>> ischar(b)
ans =
    0
```

❑ **组合字符串（strcat 和 strvcat）**

strcat 可以水平组合字符串，构成更长的字符向量。

strvcat 函数允许将不同长度的字符串组合成为字符矩阵，并且将短字符串扩充为与长字符串相同的长度。

【例 2-6】　分别使用 strcat 和 strvcat 对字符串 a 和 b 进行比较。

```
>> a='Hello';
>> b='MOTO!';
```

```
>> c=strcat(a,b)
c =
   HelloMOTO!
>> d=strvcat(a,b ,c)
d =
   Hello
   MOTO!
   HelloMOTO!
>> whos
Name     Size       Bytes  Class
  a      1x5         10    char array
  b      1x5         10    char array
  c      1x10        20    char array
  d      3x10        60    char array
Grand total is 50 elements using 100 bytes
```

❑ 比较字符串（strcmp 和 strncmp）

strcmp：比较字符串，判断字符串是否一致。

strncmp：比较字符串前 n 个字符，判断是否一致。

【例 2-7】 分别使用 strcmp 和 strncmp 对字符串 a 和 b 进行比较。

```
>> a='The first string';
>> b='The second string';
>> c=strcmp(a,b)
c =
     0
>> d=strncmp(a,b,4)
d =
     1
```

❑ 查寻索引（findstr 和 strfind）

findstr：在较长的字符串中查寻较短的字符串出现的索引。

strfind：在第一个字符串中查寻第二个字符串出现的索引。

【例 2-8】 分别使用 findstr 和 strfind 对字符串 S1 和 S2 进行查询操作。

```
>> S1='A friend in need is a friend indeed';
>> S2='friend';
>> a=findstr(S2,S1)
a =
     3    23
>> b=strfind(S2,S1)
b =
     []
>> c=strfind(S1,S2)
c =
     3    23
```

❑ **strjust** 对齐排列字符串。

【例 2-9】 对字符串 a、b、c 进行排列操作。

```
>> a='Hello';
>> b='MOTO!';
>> c=strcat(a,b)
c =
    HelloMOTO!
>> d=strvcat(a,b,c)
d =
    Hello
    MOTO!
    HelloMOTO!
>> e=strjust(d)
e =
     Hello
     MOTO!
     HelloMOTO!
```

❑ **Strrep** 替换字符串中的子串。

【例 2-10】使用 strrep 将字符串 S1 中的 firend 替换为 friend。

```
>> S1='A firend in need is a firend indeed'
S1 =
     A firend in need is a firend indeed
>> S2=strrep(S1,'firend','friend')
S2 =
     A friend in need is a friend indeed
```

❑ **Strmatch** 查询匹配的字符串。

【例 2-11】 使用 strmatch 查询字符串 a 和 b 中分别匹配 max 的字符串。

```
>> a=strmatch('max',strvcat('max','minimax','maximum'))
a =
     1
     3
>> b=strmatch('max',strvcat('max','minimax','maximum'),'exact')
b =
     1
upper 和 lower
```

5）字符串转换函数

在 MATLAB 中使用不同的函数可以允许不同类型的数据和字符串类型的数据之间进行转换；在 MATLAB 中直接提供了相应的函数对同样类型的数据进行数制的转换。

数字和字符之间的转换函数如表 2-2 所示。

表 2-2　数字和字符之间的转换函数

函　　数	说　　明
num2str	将数字转变成为字符串
int2str	将整数转变成为字符串
mat2str	将矩阵转变成为可被 eval 函数使用的字符串
str2double	将字符串转变为双精度类型的数据
str2num	将字符串转变为数字
sprinf	格式化输出数据到命令行窗口
sscanf	读取格式化字符串

不同数值之间的转换函数如表 2-3 所示。

表 2-3　不同数值之间的转换函数

函　　数	说　　明
hex2num	将十六进制整数字符串转变成为双精度数据
hex2dec	将十六进制整数字符串转变成为十进制整数
dec2hex	将十进制整数转变成为十六进制整数字符串
bin2dec	将二进制整数字符串转变成为十进制整数
dec2bin	将十进制整数转变成为二进制整数字符串
base2dec	将指定数制类型的数字字符串转变成为十进制整数
dec2base	将十进制整数转变成为指定数制类型的数字字符串

函数 str2num 在使用时需要注意，被转换的字符串仅能包含数字、小数点、字符“e”或“d”、数字的正号或负号、复数的虚部字符“i”或“j”，使用时要注意空格。

【例 2-12】 使用 str2num 函数将字符串转换为数字。

```
>> A=str2num('1+2i')
A =
   1.0000 + 2.0000i
>> B=str2num('1 +2i')
B =
   1.0000          0 + 2.0000i
>> C=str2num('1 + 2i')
C =
   1.0000 + 2.0000i
>> whos
Name   Size     Bytes    Class
  A    1x1       16     double array (complex)
  B    1x2       32     double array (complex)
  C    1x1       16     double array (complex)
Grand total is 4 elements using 64 bytes
```

也可以使用 str2double 函数避免上述问题，但 str2double 函数只能转换标量，不能转换矩阵或数组。

使用函数 num2str 将数字转换成为字符串时，可以指定字符串所表示的有效数字位数。

【例 2-13】 使用 num2str 函数将数字转换成为字符串。

```
>> A=num2str(rand(2,2),4)
A =
```

```
    0.8913    0.4565
    0.7621    0.0185
>> B=num2str(rand(2,2),6)
B =
    0.921813    0.176266
    0.738207    0.405706
```

6）格式化输入输出

MATLAB 可以进行格式化的输入、输出，用于 C 语言的格式化字符串都可以用于 MATLAB 的格式化输入输出函数，如表 2-4 所示。

表 2-4　MATLAB 的格式化输入输出函数

字　符	说　明
%c	显示内容为单一的字符
%d	有符号的整数
%e	科学计数法，使用小写的 e
%E	科学计数法，使用大写的 E
%f	浮点数据
%g	不定，在%e 或%f 之间选择一种形式
%G	不定，在%E 或%f 之间选择一种形式
%o	八进制表示
%s	字符串
%u	无符号整数
%x	十六进制表示，使用小写字符
%X	十六进制表示，使用大写字符

在 MATLAB 中，有两个函数用来进行格式化的输入和输出。

（1）sscanf（读取格式化字符串）

A=sscanf(s,format)　　　A=sscanf(s,format,size)

（2）sprintf（格式化输出数据到命令窗口）

S=sprintf(format,A,……)

【例 2-14】 分别使用 sscanf(s,format)、sscanf(s,format,size)、sprintf(format,A,……)对字符串 S1、S2、S3 进行格式化输出。

```
>> S1='2.7183 3.1416';
>> S2='2.7183e3 3.1416e3';
>> S3='0 2 4 8 16 32 64 128';
>> A=sscanf(S1,'%f')
A =
    2.7183
    3.1416
>> B=sscanf(S2,'%e')
B =
    1.0e+003*
    2.7183
    3.1416
```




```
>> C=sscanf(S3,'%d')
C =
     0
     2
     4
     8
    16
    32
    64
   128
>> S1='0 2 4 8 16 32 64 128';
>> A=sscanf(S3,'%d')
A =
     0
     2
     4
     8
    16
    32
    64
   128
>> B=sscanf(S3,'%d',1)
B =
     0
>> C=sscanf(S3,'%d',3)
C =
     0
     2
     4
>> A=1/eps;B=-eps;
>> C=[65,66,67,pi];
>> D=[pi,65,66,67];
>> S1=sprintf('%+15.5f',A)
S1 =
    +4503599627370496.00000
>> S2=sprintf('%+.5e',B)
S2 =
    -2.22045e-016
>> S3=sprintf('%s%f',C)
S3 =
    ABC3.141593
>> S4=sprintf('%s%f%s',D)
S4 =
    3.141593e+00065.000000BC
```

- ❑ 格式化字符串中若包含了“+”，则表示在输出的字符串中包含数据的符号。
- ❑ 对于整数数值进行格式化输出时，可以直接将向量转变成为字符串。
- ❑ 如果输出的数据与相应的格式化字符串不匹配，则输出为数值最常见的形式。
- ❑ MATLAB 提供了 input 函数来完成获取用户输入数据的功能，以满足能够和用户的输入进行交互的需要。

A=input(prompt)：参数 prompt 为提示用的字符串。

A=input(prompt,'s')：若有 s，则输入的数据为字符串；没有 s，则输入的数据为双精度数据。

【例 2-15】 input 函数的使用方法。

```
>> A=input('随便输入数字：')
随便输入数字：264
A =
   264
>> B=input('随便输入数字：','s')
随便输入数字：264
B =
   264
>> whos
 Name      Size              Bytes    Class
 A         1x1                8       double array
 B         1x3                6       char array
Grand total is 4 elements using 14 bytes
```

2.1.2 数值（Numeric）

MATLAB 的基本数值类型变量或对象主要用来描述基本的数值对象。

MATLAB 还存在的一些数据：常量数据、空数组或空矩阵等。常量数据是指在使用 MATLAB 过程中由 MATLAB 提供的公共数据，数据可以通过数据类型转换的方法转换常量到不同的数据类型，还可以被赋予新的数值；空数组或空矩阵，在创建数组或矩阵时，可以使用空数组或空矩阵辅助创建数组或矩阵。

1. 基本数值类型

基本数据类型如表 2-5 所示。

表 2-5 基本数值类型

数据类型	说明	字节数
double	双精度数据类型	8
sparse	稀疏矩阵数据类型	N/A
single	单精度数据类型	4
uint8	无符号 8 位整数	1
uint16	无符号 16 位整数	2
uint32	无符号 32 位整数	4
uint64	无符号 64 位整数	8
int8	有符号 8 位整数	1
int16	有符号 16 位整数	2
int32	有符号 32 位整数	4
int64	有符号 64 位整数	8

class 函数可以用来获取变量或对象的类型，也可以用来创建用户自定义的数据类型。

【例 2-16】 class 函数的使用样例。

```
>> A=[1 2 3];
>> class(A)
ans =
     double
>> whos
  Name      Size            Bytes       Class
  A          1x3             24          double array
  ans        1x6             12          char array
Grand total is 9 elements using 36 bytes
>> B=int16(A);
>> class(B)
ans =
     int16
>> whos
 Name      Size              Bytes      Class
 A          1x3               24         double array
 B          1x3                6          int16 array
 ans        1x5               10         char array
Grand total is 11 elements using 40 bytes
```

📖 MATLAB 和 C 语言在处理数据类型和变量时的区别：在 C 语言中，任何变量在使用之前必须声明，然后赋值，在声明变量时就指定了变量的数据类型；在 MATLAB 中，任何数据变量都不需要预先声明，MATLAB 将自动将数据类型设置为双精度类型。

- ❑ MATLAB 系统默认的运算都是针对双精度类型的数据或变量；
- ❑ 稀疏矩阵的元素仅能使用双精度类型的变量；
- ❑ Spares 类型的数据变量和整数类型数据、单精度数据类型变量之间的转换是非法的。

在进行数据类型转换时，若输入参数的数据类型就是需要转换的数据类型，则 MATLAB 忽略转换，保持变量的原有特性。

2. 整数类型数据运算

整数类型数据的运算函数如表 2-6 所示。

表 2-6 整数类型数据的运算函数

函　数	说　明
bitand	数据位“与”运算
bitcmp	按照指定的数据位数求数据的补码
bitor	数据位“或”运算
bitmax	最大的浮点整数数值
bitxor	数据位“异或”运算
bitset	将指定的数据位设置为 1
bitget	获取指定的数据位数值
bitshift	数据位移操作



参与整数运算的数据都必须大于 0。

❑ **bitand** 数据位“与”操作函数。

【例 2-17】 使用 bitand 函数对数据 A 和 B 进行与操作。

```
>> A=86;B=77;
>> C=bitand(A,B)
C =
    68
>> a=uint16(A);b=uint16(B);
>> c=bitand(a,b)
c =
    68

86 的补码:          01010110
77 的补码:          01001101
"与"运算的结果:  01000100
>> whos
  Name      Size            Bytes      Class
  A         1x1                 8      double array
  B         1x1                 8      double array
  C         1x1                 8      double array
  a         1x1                 2      uint16 array
  b         1x1                 2      uint16 array
  c         1x1                 2      uint16 array
Grand total is 6 elements using 30 bytes
```

❑ **bitset** 将指定的数据位设置为“1”函数。

【例 2-18】 使用 bitset 函数对数据 A 进行操作。

```
>> A=86;
>> dec2bin(A)
ans =
     1010110
>> B=bitset(A,6)
B =
   118
>> dec2bin(B)
ans =
     1110110
>> C=bitset(A,7,0)
C =
    22
>> dec2bin(C)
ans =
     10110
```

📖 bitset(A,B,C)函数根据输入的第二个参数设置相应的数据位的数值，若不指定第三个参数，则将相应的数据位设置为“1”，否则，根据输入的第三个参数设置相应的数据位。

❑ **bitget** 数据位操作函数，用于获取指定的数据位数值。

用法：bitget(A,B)，根据输入的第二个参数获取指定的数据位的数值。

【例 2-19】 使用 bitget 函数获取数据 A 指定数据位的值。

```
>> A=86;
>> dec2bin(A)
ans =
    1010110
>> bitget(A,6)
ans =
    0
>> bitget(A,3)
ans =
    1
>> A=86;
>> bitget(A,6)
ans =
    0
>> bitget(A,3)
ans =
    1
```

❑ **bitshift** 数据位移动操作函数。

用法：bitshift(A,B)，函数第二个参数为正，则左移，第二个参数为负，则右移。

【例 2-20】 使用 bitshift 函数对数据进行移位操作。

```
>> A=86;
>> dec2bin(A)
ans =
     1010110
>> D=bitshift(A,4);
>> dec2bin(D)
ans =
     10101100000
>> E=bitshift(A,-4);
>> dec2bin(E)
ans =
     101  →(0000101)
>> A=86;
>> D=bitshift(A,4)
D =
     1376
>> E=bitshift(A,-4)
E =
    5
```

3. MATLAB 的常量

MATLAB 的常量如表 2-7 所示。

表 2-7　MATLAB 的常用常量

常　　量	说　　明
ans	最近运算的结果
eps	浮点数相对精度，定义为 1.0 到最近浮点数的距离
realmax	MATLAB 能表示的实数的最大绝对值
realmin	MATLAB 能表示的实数的最小绝对值
pi	圆周率的近似值 3.1415926
i，j	复数的虚部数据最小单位
inf 或 Inf	表示正无限大，定义为 1/0
NaN 或 nan	非数，它产生于 0×（空格），0/0，（空格）/（空格）等运算


注意

eps、realmax、realmin 三个常量具体的数值与运行 MATLAB 的计算机相关，不同的计算机系统可能具有不同的数值。

MATLAB 的常量是可以赋予性的数值的，一旦被赋予了新的数值，则常量代表的就是新值，而不是原有的值，只有执行 clear 命令后，常量才会代表原来的值，下例将进行演示。

【例 2-21】 使用 clear 命令将 pi 恢复原值。

```
>> pi=100
pi =
  100
>> clear
>> pi
ans =
   3.1416
```

将 inf 应用于函数，计算结果可能为 inf 或 NaN。进行数据转换时，inf 将获取相应数据类型的最大值，而 NaN 返回相应整数数据类型的数值 0，浮点数类型则仍然为 NaN。

【例 2-22】 inf 或 NaN 的使用样例。

```
>> A=Inf;
>> class(A)
ans =
     Double
>> B=int16(A)
B =
   32767
>> C=sin(A)
C =
   NaN
>> sin(C)
ans =
   NaN
>> class(C)
ans =
     double
>> int64(C)
```

```
ans =
     0
>> int32(C)
ans =
     0
```

4. 空数组

空数组不意味着什么都没有，空数组类型的变量在 MATLAB 的工作空间中是存在的。

【例 2-23】 创建空数组。

```
>> A=[]
A =
   []
>> B=ones(2,3,0)
B =
   Empty array: 2-by-3-by-0
>> C=randn(2,3,4,0)
C =
   Empty array: 2-by-3-by-4-by-0
>> whos
  Name      Size           Bytes    Class
  A         0x0              0      double array
  B         2x3x0            0      double array
  C         4-D              0      double array
Grand total is 0 elements using 0 bytes
```

使用空数组，可以将大数组删除部分行或列，亦可以删除多维数组的某一页。

【例 2-24】 使用空数组对大数组进行列删除操作。

```
>> A=reshape(1:24,4,6)
A =
     1     5     9    13    17    21
     2     6    10    14    18    22
     3     7    11    15    19    23
     4     8    12    16    20    24
>> %删除第 2.3.4 列
>> A(:,[2 3 4])=[]
A =
     1    17    21
     2    18    22
     3    19    23
     4    20    24
```

思考：如何删除第 2、3 行？

```
>>  ? ? ?
A =
1    5    9    13    17    21
4    8    12   16    20    24
```

2.1.3 函数句柄（Handle）

函数句柄(Function handle)是MATLAB的一种数据类型。引入函数句柄是为了使feval及借助于它的泛函指令工作更可靠；特别在反复调用情况下更显效率；使“函数调用”像“变量调用”一样方便灵活；提高函数调用速度，提高软件重用性，扩大子函数和私用函数的可调用范围；迅速获得同名重载函数的位置、类型信息。MATLAB中函数句柄的使用使得函数也可以成为输入变量，并且能很方便地调用，提高函数的可用性和独立性。

函数句柄可以将其理解成一个函数的代号，就像一个人的名字，这样在调用时可以调用函数句柄而不用调用该函数。

创建函数句柄需要用到操作符@，创建函数句柄的语法如下。

```
fhandle = @function_filename
```

调用函数时就可以调用该句柄，可以实现同样的功能。

例如，

```
fhandle = @sin
```

就创建了sin的句柄，输入fhandle(x)其实就是sin(x)的功能。

2.1.4 逻辑（Logical）类型和关系运算

逻辑运算又称布尔运算。布尔用数学方法研究逻辑问题，成功地建立了逻辑演算。它用等式表示判断，把推理看作等式的变换，这种变换的有效性不依赖人们对符号的解释，只依赖于符号的组合规律，这一逻辑理论常被称为布尔代数。20世纪30年代，逻辑代数在电路系统上获得应用，随后，由于电子技术与计算机的发展，出现各种复杂的大系统，其变换规律也遵守布尔所揭示的规律。逻辑运算（logical operators）通常用来测试真假值。最常见的逻辑运算就是循环处理，用来判断是否该离开循环或继续执行循环内的指令。

关系的基本运算有两类：一类是传统的集合运算（并、差、交等）；另一类是专门的关系运算（选择、投影、连接、除法、外连接等），有些查询需要几个基本运算的组合，要经过若干步骤才能完成。

1．逻辑数据类型

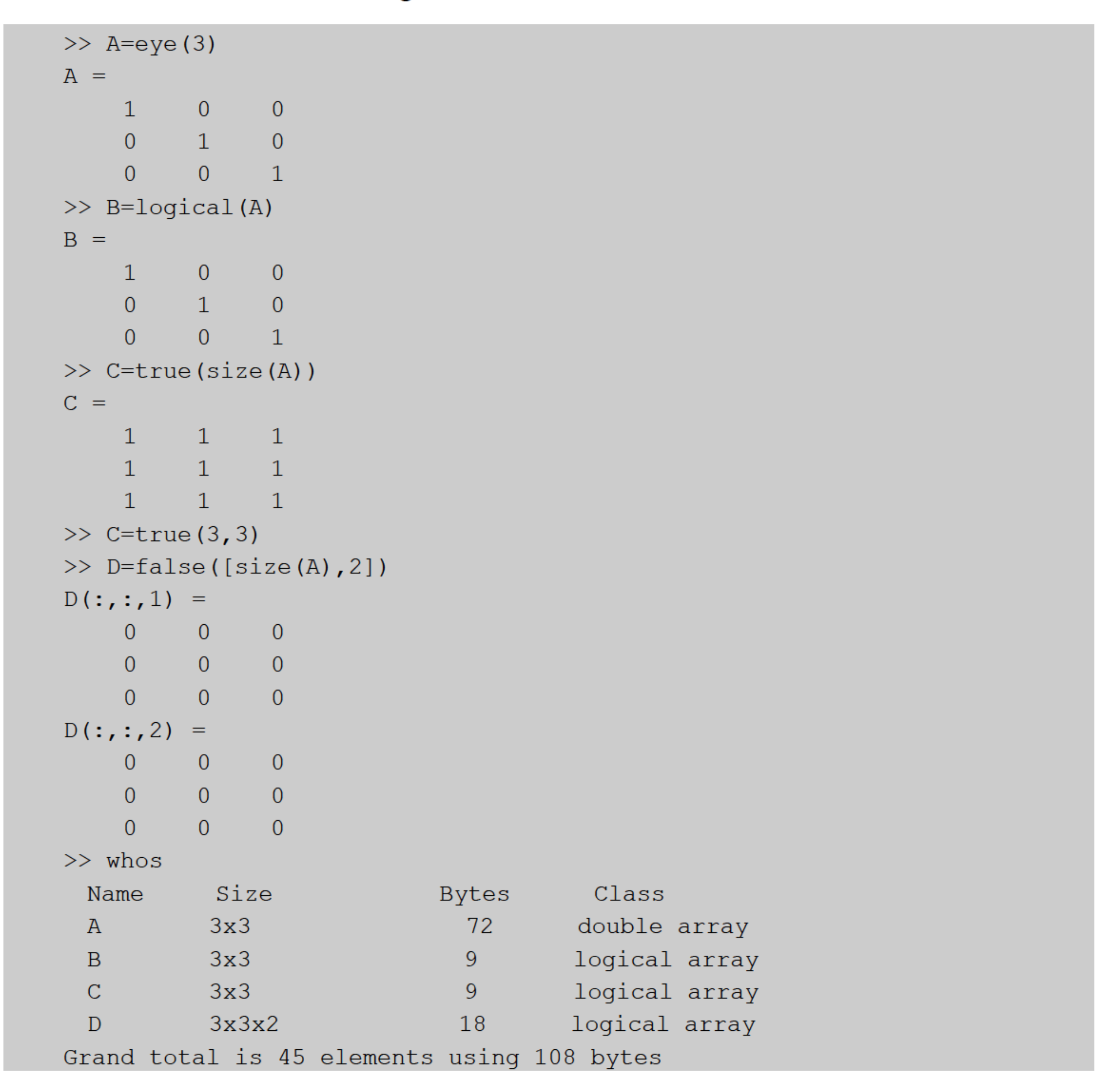
在MATLAB中逻辑类型包含true和false，分别由1和0表示。在MATLAB中用函数logical()将任何非零的数值转换为true（即1），将数值0转换为false（即0）。逻辑类型的数据只能通过数值类型转换，或者使用特殊的函数生成相应类型的数组或矩阵。逻辑类型的数组每一个元素仅占用一个字节的内存空间。

创建逻辑类型数据的函数如表2-8所示。

表 2-8　创建逻辑类型数据的函数

函　　数	说　　明
logical	将任意类型的数组转变为逻辑类型数组，其中非零元素为真，零元素为假
True	产生逻辑真值数组
False	产生逻辑假值数组

【例 2-25】 逻辑数据类型 logical、True、False 的使用样例。



```
>> A=eye(3)
A =
     1     0     0
     0     1     0
     0     0     1
>> B=logical(A)
B =
     1     0     0
     0     1     0
     0     0     1
>> C=true(size(A))
C =
     1     1     1
     1     1     1
     1     1     1
>> C=true(3,3)
>> D=false([size(A),2])
D(:,:,1) =
     0     0     0
     0     0     0
     0     0     0
D(:,:,2) =
     0     0     0
     0     0     0
     0     0     0
>> whos
  Name      Size              Bytes     Class
  A         3x3                 72     double array
  B         3x3                 9      logical array
  C         3x3                 9      logical array
  D         3x3x2               18     logical array
Grand total is 45 elements using 108 bytes
```

在使用 true 或 false 函数创建逻辑类型数组时，若不指明参数，则创建一个逻辑类型的标量。

在 MATLAB 中有些函数以 is 开头，这类函数用来完成某种判断功能的函数。

例如，

- **isnumeric（*）**　判断输入的参数是否为数值类型。
- **islogical（*）**　判断输入的参数是否为逻辑类型。

【例 2-26】 isnumeric 与 islogical 的使用方法。

```
>> a=true
a =
     1
>> b=false
b =
     0
>> c=1
c =
     1
>> isnumeric(a)
ans =
     0
>> isnumeric(c)
ans =
     1
>> islogical(a)
ans =
     1
>> islogical(b)
ans =
     1
>> islogical(c)
ans =
     0
```

2. 逻辑运算

能够处理逻辑类型数据的运算叫逻辑运算。参与逻辑运算的操作数不一定是逻辑类型的变量或常量，其他类型的数据也可以进行逻辑运算，但运算结果一定是逻辑类型的数据。

MATLAB 的逻辑运算符及其作用如表 2-9 所示。

表 2-9 MATLAB 的逻辑运算符及其作用

运 算 符	说 明
& &	具有短路作用的逻辑与操作，仅能处理标量
| |	具有短路作用的逻辑或操作，仅能处理标量
&	元素与操作
|	元素或操作
~	逻辑非操作
xor	逻辑异或操作
any	当向量中的元素有非零元素时，返回真
all	当向量中的元素都是非零元素时，返回真

具有短路作用的逻辑“与”操作（&&）和“或”操作（||），在进行 a && b && c && d 运算时，若 a 为假（0），则后面的三个变量都不再被处理，运算结束，并返回运算结果逻辑假（0）；同样，进行 a || b || c || d 运算时，若 a 为真（1），则后面的三个变量都不再被处理，运算结束，并返回运算结果逻辑真（1）。



它们仅能处理标量！

【例 2-27】 对数据 a、b、c、d 进行“与”操作（&&）和“或”操作（||）。

```
>> a=0;b=1;c=2;d=3;
>> a&&b&&c&&d
ans =
     0
>> a=0;b=2;c=6;d=8;
>> a&&b&&c&&d
ans =
     0
>> a=10;b=1;c=2;d=3;
>> a||b||c||d
ans =
     1
>> a=10;b=0;c=7;d=9;
>> a||b||c||d
ans =
     1
>> whos
  Name      Size            Bytes  Class
  a          1x1              8     double array
  ans        1x1              1     logical array
  b          1x1              8     double array
  c          1x1              8     double array
  d          1x1              8     double array
Grand total is 5 elements using 33 bytes
```

函数 any 和 all 是针对矩阵中每一列进行处理。Any 对每列元素有非零值，则返回逻辑真；All 对每列元素均为非零值，则返回逻辑真。

【例 2-28】 使用 any 和 all 分别对数据 a、b、c 进行操作。

```
>> a=[1 2 3 0];
>> any(a)
ans =
     1
>> all(a)
ans =
     0
>> b=[0 0 0 0];
>> any(b)
ans =
     0
>> all(b)
ans =
     0
>> c=[1 2 3 4];
>> any(c)
```

```
ans =
     1
>> all(c)
ans =
     1
>> a=[1 0 2;3 0 0;1 3 0;1 1 1]
a =
     1     0     2
     3     0     0
     1     3     0
     1     1     1
>> any(a)
ans =
     1     1     1
>> all(a)
ans =
     1     0     0
```

3. 关系运算

MATLAB 的关系运算符如表 2-10 所示。

表 2-10　MATLAB 的关系运算符

运　算　符	说　　明
==	等于
~=	不等于
<	小于
>	大于
<=	小于等于
>=	大于等于

参与关系运算的操作数可以是各种数据类型的变量或常数，其运算结果是逻辑类型的数据，标量可以和数组（或矩阵）进行比较，比较时自动扩展标量，返回的结果是和数组同维的逻辑类型数组，若比较的是两个数组，则数组必须是同维的，且每一维的尺寸必须一致。

利用“()”和各种运算符相结合，可以完成复杂的关系运算。

【例 2-29】 灵活运用“()”进行运算操作。

```
>> A=reshape(-4:4,3,3)
A =
    -4    -1     2
    -3     0     3
    -2     1     4
>> A>=0
ans =
     0     0     1
     0     1     1
     0     1     1
>> B=~(A>=0)
```

```
B =
     1     1     0
     1     0     0
     1     0     0
>> whos
Name      Size      Bytes   Class
  A        3x3         72      double array
  B        3x3         9       logical array
  ans      3x3         9       logical array
Grand total is 27 elements using 90 bytes
>> C=(A>0)&(A<3)
C =
     0     0     1
     0     0     0
     0     1     0
>> A>0
ans =
     0     0     1
     0     0     1
     0     1     1
>> A<3
ans =
     1     1     1
     1     1     0
     1     1     0
A =
    -4    -1     2
    -3     0     3
    -2     1     4
```

逻辑索引：将逻辑类型的数据应用于索引就构成了逻辑索引，利用逻辑索引可以方便地从矩阵或数组中找到某些符合条件的元素。

❑ **运算符的优先级**

（1）括号（）

（2）数组转置（. ），数组幂（.^ ），矩阵转置（ ’ ），矩阵幂（^）

（3）一元加（+），一元减（−），逻辑非（~）

（4）数组乘法（.＊ ），数组右除（./ ），数组左除（.\ ），矩阵乘法（＊），矩阵右除（/），矩阵左除（\）

（5）加法（+），减法（−）

（6）冒号运算符（ ：）

（7）小于（<），小于等于（<=），大于（>），大于等于（>=），等于（==），不等于（~=）

（8）元素与（&）

（9）元素或（|）

（10）短路逻辑与（&&）

（11）短路逻辑或（||）

2.1.5 结构体（Structure）类型

结构是包含一组记录的数据类型，记录是存储在相应的字段中，结构的字段可以是任意一种 MATLAB 数据类型的变量或对象，结构类型的变量可以是一维的、二维的或多维的数组，在访问结构类型数据的元素时，需要使用下标配合字段的形式。

1．结构的创建

结构的创建有两种方法：直接赋值和利用 struct 函数创建。

（1）直接赋值创建结构

创建时直接用结构的名称，配合操作符“.”和相应的字段名称完成创建，创建是直接给字段赋具体的数值。

【例 2-30】 Student 结构的创建。

```
>> Student.name='Way';
>> Student.age=26;
>> Student.grade=uint16(1);
>> whos
  Name        Size              Bytes   Class
Student      1x1                388     struct array
Grand total is 8 elements using 388 bytes
>> Student
Student =
     name: 'Way'
  age: 26
     grade: 1
```

MATLAB 会自动扩展结构数组的尺寸，对于没有赋值的字段，则直接创建空数组。

【例 2-31】 空结构数组的创建。

```
>> Student(2)
ans =
     name: []
      age: []
    grade: []
>> Student(3).age
ans =
     []
```

（2）利用 struct 函数创建结构

struct 函数的基本语法如下。

- struct-name= struct(field1,val1,field2,val2,……)
- struct-name= struct(field1,{val1},field2,{val2},……)

【例 2-32】 使用 struct 函数创建 Student 结构。

```
>> Student=struct('name','Way','age',26,'grade',uint16(1))
Student =
     name: 'Way'
```

```
  age: 26
    grade: 1
>> whos
Name        Size    Bytes       Class
Student     1x1     388         struct array
Grand total is 8 elements using 388 bytes

>> Student=struct('name',{'Deni','Sherry'},'age',{22,24},'grade',{2,3})
Student =
1x2 struct array with fields:
    name
    age
    grade
>> whos
  Name          Size            Bytes         Class
  Student       1x2             604           struct array
Grand total is 20 elements using 604 bytes

>> Student=struct('name',{},'age',{},'grade',{})
Student =
0x0 struct array with fields:
    name
    age
    grade
>> whos
  Name          Size                Bytes    Class
  Student       0x0                  192     struct array
Grand total is 0 elements using 192 bytes
```

可以使用 repmat 函数，给结构制作复本。

【例 2-33】 使用 repmat 函数给 Student 结构制作复本。

```
>>Student=repmat(struct('name','Way','age',26,'grade',uint16(1)),1,2)
Student =
1x2 struct array with fields:
    name
    age
    grade
>> Student=repmat(struct('name','Way','age',26,'grade',uint16(1)),1,3)
Student =
1x3 struct array with fields:
    name
    age
    Grade

>> Student(1)
ans =
     name: 'Way'
      age: 26
    grade: 1
>> Student(2)
```

```
ans =
     name: 'Way'
      age: 26
    grade: 1
```

2. 结构的基本操作

对于结构的基本操作其实是对结构数组元素包含记录的操作，其中，包括结构记录数据的访问和字段的增加和删除。

（1）访问结构数组元素包含的记录的方法。

可以直接使用结构数组的名称和字段的名称及操作符“.”完成相应的操作，也可以使用“动态”字段的形式利用动态字段形式访问结构数组元素，便于利用函数完成对结构字段数据的重复操作。

基本语法结构如下。

❑ **struct-name.(expression)**

【例 2-34】 直接使用结构数组的名称（Student）和字段的名称访问其中的元素。

```
>>Student=struct('name',{'Deni','Sherry'},'age',{22,24},'grade',{2,3},
               'score',{rand(3)*10,randn(3)*10});
>> Student
Student =
1x2 struct array with fields:
    name
    age
    grade
    score
>> Student(2).score
ans =
   -4.3256    2.8768   11.8916
  -16.6558  -11.4647   -0.3763
    1.2533   11.9092    3.2729
```

利用动态字段的形式可以通过编写函数对记录的数据进行统一的运算操作。

【例 2-35】 使用动态字段对 Student 中的数据进行统一操作。

```
>> Student(2).score(1,:)
ans =
   -4.3256    2.8768   11.8916
>> Student.name
ans =
     Deni
ans =
     Sherry
>> Student.('name')
ans =
     Deni
ans =
     Sherry
```

（2）对结构数据进行计算。

若对结构数组的某一个元素的字段代表的数据进行计算，则和使用 MATLAB 普通的变量操作相同；若对结构数组的某一个字段的所有数据进行同一种操作，则需要使用[]符号将该字段包含起来。

【例 2-36】 对 Student 的数据求平均值。

```
>> mean(Student(1).score)      % Mean 函数是用来求解列向量的平均值
ans =
    6.1736    6.1210    7.5269
>> mean([Student.score])
ans =
    6.1736    6.1210    7.5269   -6.5761    1.1071    4.9294
```

（3）内嵌结构。

当结构的字段记录了结构时，则称其为内嵌结构，创建内嵌结构可以使用直接赋值的方法，也可以使用 struct 函数完成。

【例 2-37】 使用直接赋值的方法创建内嵌结构。

```
>>Student=struct('name',{'Deni','Sherry'},'age',{22,24},
               'grade',{2,3},'score',{rand(3)*10,randn(3)*10});
>> Class.numble=1;
>> Class.Student=Student;
>> whos
Name          Size            Bytes      Class
Class          1x1             1188       struct array
Student        1x2              932       struct array
Grand total is 83 elements using 2120 bytes
>> Class
Class =
    numble: 1
Student: [1x2 struct]
```

【例 2-38】 使用 struct 函数创建内嵌结构。

```
>>Class=struct('numble',1,'Student',struct('name',{'Way','Deni'}))
Class =
    numble: 1
Student: [1x2 struct]
```

（4）结构操作函数，如表 2-11 所示。

表 2-11　结构操作函数

函　　数	说　　明
struct	创建结构或其他数据类型转变成结构
fieldnames	获取结构的字段名称
getfield	获取结构字段的数据
setfield	设置结构字段的数据
rmfield	删除结构的指定字段

续表

函　　数	说　　明
isfield	判断给定的字符串是否为结构的字段名称
isstruct	判断给定的数据对象是否为数据类型
oderfields	将结构字段排序

❑ **setfield 函数**　设置结构字段的数据。

❑ **fieldnames 函数**　获取结构的字段名称。

【例 2-39】 使用 fieldnames 函数获取 S 结构的字段名称。

```
>> fieldnames(S)
ans =
    'name'
    'ID'
```

❑ **getfield 函数**　获取结构字段的数据。

【例 2-40】 用 getfield 函数获取 S 结构的字段数据。

```
>> A=getfield(S,{2,2},'name')
A =
   Way
>> B=getfield(S,{2,2},'ID')
B =
   1
```

❑ **orderfields 函数**　将结构字段排序，该函数能够将结构的字段按照字符序号排列。

【例 2-41】 用 orderfields 函数对 S3 结构字段排序。

```
>> S3=orderfields(S)
S3 =
2x2 struct array with fields:
    ID
name
```

❑ **rmfield 函数**　删除结构的指定字段。

【例 2-42】 用 rmfield 函数删除 S4 的 ID 字段。

```
>> S4=rmfield(S,'ID')
S4 =
2x2 struct array with fields:
    name
```

❑ **isfield 函数**　判断给定的字符串是否为结构的字段名称。

【例 2-43】 使用 isfield 函数判断 name 和 id 字段是否分别属于结构 A 和 B。

```
>> A=isfield(S,'name')
A =
     1
>> B=isfield(S,'id')
```

```
B =
     0
```

❑ **isstruct 函数**　判断给定的数据对象是否为结构类型。

【例 2-44】 使用 isstruct 函数判断数据 S 是否为结构类型。

```
>> isstruct(S)
ans =
     1
```

❑ **cell2struct 函数**　将元胞数组转变成为结构。

❑ **struct2cell 函数**　将结构转变成为元胞数组。

❑ **deal 函数**

deal 函数处理标量时，将标量的数值依次赋值给相应的输出。

【例 2-45】 使用 deal 函数依次给 Y1、Y2、Y3 赋值。

```
>> X=3;
>> [Y1,Y2,Y3]=deal(X)
Y1 =
     3
Y2 =
     3
Y3 =
     3
```

deal 函数处理元胞数组时，将元胞数组中的元胞依次赋值给相应的输出。

【例 2-46】 使用 deal 函数依次给元胞数组赋值并输出。

```
>> X={rand(3),'2',1};
>> [Y1,Y2,Y3]=deal(X{:})
Y1 =
    0.9501    0.4860    0.4565
    0.2311    0.8913    0.0185
    0.6068    0.7621    0.8214
Y2 =
    2
Y3 =
    1
```

2.1.6　元胞数组（cell）类型

元胞数组是 MATLAB 的一种特殊数据类型，可以将元胞数组看作为一种无所不包的通用矩阵（广义矩阵），组成元胞数组的元素可以是任何一种数据类型的常数或常量。数据类型可以是字符串、双精度数、稀疏矩阵、元胞数组、结构或其他 MATLAB 数据类型。每一个元胞数据可以是标量、向量、矩阵、N 维数组，每一个元素可以具有不同的尺寸和内存空间，每一个元素的内容可以完全不同，元胞数组的元素叫作元胞。元胞数组的内存空间是动态分配的，它的维数不受限制。访问元胞数组的元素可以使用单下标方式或全下

标方式。

元胞数组和结构数组的异同，如表 2-12 所示。

表 2-12　元胞数组和结构数组的异同

内　　容	元胞数组对象	结构数组对象
基本元素	元胞	结构
基本索引	全下标方式、单下标方式	全下标方式、单下标方式
可包含的数据类型	任何数据类型	任何数据类型
数据的存储	元胞	字段
访问元素的方法	花括号和索引	圆括号、索引和字段名

1. 元胞数组的创建

（1）使用运算符花括号{}，将不同类型和尺寸的数据组合在一起构成一个元胞数组。

【例 2-47】 构造元胞数组 A。

```
>> A={zeros(2,2,2),'Hello';17.35,1:100}
A =
    [2x2x2 double]    'Hello'
    [  17.3500]    [1x100 double]
>> whos
  Name    Size    Bytes    Class
  A       2x2     1122     cell array
Grand total is 118 elements using 1122 bytes
```

对于内容较多的元胞，显示的内容将为元胞的数据类型和尺寸。

（2）将数组的每一个元素用{}括起来，然后用数组创建的符号[]将数组的元素括起来构成一个元胞数组。

【例 2-48】 创建由数组元素构成的元胞数组。

```
>> B=[{zeros(2,2,2)},{'Hello'};{17.35},{1:100}]
B =
    [2x2x2 double]    'Hello'
    [  17.3500]    [1x100 double]
>> whos
  Name     Size     Bytes     Class
  A        2x2      1122      cell array
Grand total is 118 elements using 1122 bytes
```

（3）用{}创建一个元胞数组，MATLAB 能够自动扩展数组的尺寸，没有明确赋值的元素作为空元胞数组存在。

【例 2-49】 用{}创建一个元胞数组。

```
>> C={1}
C =
    [1]
>> whos
  Name     Size     Bytes     Class
  C       1x1      68       cell array
```

```
Grand total is 2 elements using 68 bytes
>> C(2,2)={3}
C =
    [1]     []
     []    [3]
>> whos
  Name    Size    Bytes    Class
  C        2x2      144    cell array
Grand total is 6 elements using 144 bytes
```

（4）用函数 cell 创建元胞数组，该函数可以创建一维、二维或多维元胞数组，但创建的数组都为空元胞。

【例 2-50】 用函数 cell 创建元胞数组。

```
>> A=cell(1)
A =
    {[]}
>> B=cell(2,3)
B =
     []     []     []
     []     []     []
>> C=cell(2,2,2)
C(:,:,1) =
     []     []
     []     []
C(:,:,2) =
     []     []
     []     []
>> whos
  Name      Size              Bytes     Class
  A          1x1                 4     cell array
  B          2x3                24     cell array
  C          2x2x2              32     cell array
Grand total is 15 elements using 60 bytes
```

元胞数组的每个空元胞占用四个字节的内存空间，元胞数组占用的内存空间和元胞数组的内容有关，不同的元胞数组占用的内存空间不同。

2. 元胞数组的基本操作

元胞数组的基本操作包括对元胞数组元胞和元胞数据的访问、修改，元胞数组的扩展、收缩或重组。操作数值数组的函数也可以应用在元胞数组上。

（1）元胞数组的访问

使用圆括号()直接访问元胞数组的元胞，获取的数据也是一个元胞数组。

【例 2-51】 使用圆括号（）直接访问元胞数组的元胞。

```
>> A=[{zeros(2,2,2)},{'Hello'};{17.35},{1:100}]
A =
    [2x2x2 double]    'Hello'
    [17.3500]    [1x100 double]
>> B=A(1,2)
```

```
B =
    'Hello'
>> class(B)
ans =
     Cell
>> whos
Name      Size          Bytes      Class
  A        2x2           1122     cell array
  B        1x1             70     cell array
  ans      1x4              8     char array
Grand total is 128 elements using 1200 bytes
```

使用花括号{}直接访问元胞数组的元胞，获取的数据是字符串。

【例 2-52】 使用花括号{}直接访问元胞数组的元胞。

```
>> A=[{zeros(2,2,2)},{'Hello'};{17.35},{1:100}]
A =
    [2x2x2 double]    'Hello'
    [     17.3500]    [1x100 double]
>> C=A{1,2}
C =
    Hello
>> class(C)
ans =
    Char
>> whos
Name      Size              Bytes     Class
  A        2x2              1122      ell array
  C        1x5                10      char array
  ans      1x4                 8      char array
Grand total is 127 elements using 1140 bytes
```

【例 2-53】 将花括号{}和圆括号( )结合起来使用访问元胞元素内部的成员。

```
>> A=[{zeros(2,2,2)},{'Hello'};{17.35},{1:10}]
A =
    [2x2x2 double]    'Hello'
    [     17.3500]    [1x10 double]
>> D=A{1,2}(2)
D =
    E
>> E=A{2,2}(5:end)
E =
    5     6     7     8     9    10
>> class(E)
ans =
     Double
>> F=A{4}([1 3 5])
F =
     1     3     5
>> whos
```

```
  Name      Size            Bytes     Class
  A         2x2              402      cell array
  D         1x1                2      char array
  E         1x6               48      double array
  F         1x3               24      double array
  ans        1x6              12      char array
Grand total is 44 elements using 488 bytes
```

（2）元胞数组的扩充（其方法和数值数组大体相同）

【例 2-54】 元胞数组的扩充样例。

```
>> A=[{zeros(2,2,2)},{'Hello'};{17.35},{1:10}]
A =
    [2x2x2 double]    'Hello'
    [        17.3500]    [1x10 double]
>> B=cell(2)
B =
     []     []
     []     []
>> B(:,1)={char('Hello','Welcome');10:-1:5}
B =
    [2x7 char   ]     []
    [1x6 double]     []
>> C=[A,B]
C =
    [2x2x2 double]    'Hello'          [2x7 char   ]      []
    [        17.3500]  [1x10 double]    [1x6 double]      []
>> D=[A,B;C]
D =
    [2x2x2 double]    'Hello'           [2x7 char   ]     []
    [        17.3500]  [1x10 double]     [1x6 double]     []
    [2x2x2 double]    'Hello'           [2x7 char   ]     []
    [        17.3500]  [1x10 double]     [1x6 double]     []
>> whos
  Name       Size          Bytes           Class
  A         2x2            402           cell array
  B         2x2            204           cell array
  C         2x4            606           cell array
  D         4x4           1212           cell array
Grand total is 208 elements using 2424 bytes
```

（3）元胞数组的收缩和重组（和数值数组大体相同）

【例 2-55】 元胞数组的收缩。

```
D =
    [2x2x2 double]    'Hello'          [2x7 char   ]      []
    [        17.3500]  [1x10 double]    [1x6 double]      []
    [2x2x2 double]    'Hello'          [2x7 char   ]      []
    [        17.3500]  [1x10 double]    [1x6 double]      []
>> D(2,:)=[]
D =
```

```
    [2x2x2 double]    'Hello'              [2x7 char ]     []
    [2x2x2 double]    'Hello'              [2x7 char ]     []
    [         17.3500]  [1x10 double]      [1x6 double]     []
```

【例 2-56】 元胞数组的重组。

```
>> E=reshape(D,2,2,3)
E(:,:,1) =
    [2x2x2 double]    [17.3500]
    [2x2x2 double]    'Hello'
E(:,:,2) =
    'Hello'           [2x7 char]
    [1x10 double]     [2x7 char]
E(:,:,3) =
    [1x6 double]     [ ]
     [ ]             [ ]
```

（4）元胞数组的操作函数，如表 2-13 所示。

表 2-13　元胞数组的操作函数

函　　数	说　　明
cell	创建空的元胞数组
cellfun	为元胞数组的每个元胞执行指定的函数
celldisp	显示所有元胞的内容
cellplot	利用图形方式显示元胞数组
cell2mat	将元胞数组转变成为普通的矩阵
mat2cell	将普通的值矩阵转变成为元胞数组
num2cell	将数值数组转变成为元胞数组
deal	将输入参数赋值给输出
cell2struct	将元胞数组转变成为结构
struct2cell	将结构转变成为元胞数组
iscell	判断输入是否为元胞数组

- **cellfun 函数**　主要功能是对元胞数组的元素(元胞)分别指定不同的函数，在 cellfun 函数中可用的函数如表 2-14 所示。

表 2-14　在 cellfun 函数中可用的函数

函　　数	说　　明
isempty	若元胞元素为空，则返回逻辑真
islogical	若元胞元素为逻辑类型，则返回逻辑真
isreal	若元胞元素为实数，则返回逻辑真
length	元胞元素的长度
ndims	元胞元素的维数
prodofsize	元胞元素包含的元素个数

【例 2-57】 对元胞数组的元素（元胞）分别指定不同的函数。

```
>> A={rand(2,2,2),'Hello',pi;17,1+i,magic(5)}
A =
```

```
    [2x2x2 double]    'Hello'                [  3.1416   ]
    [            17]    [1.0000+ 1.0000i]    [5x5 double]
>> B=cellfun('isreal',A)
B =
     1     1     1
     1     0     1
>> C=cellfun('length',A)
C =
     2     5     1
     1     1     5
```

cellfun 函数还有以下两种用法。

❑ **cellfun（'size',C,K）** 获取元胞数组元素第 K 维的尺寸。

❑ **cellfun（'isclass',C,classname）** 判断元胞数组的数据类型。

【例 2-58】 获取元胞数组 A 元素第 1 维的尺寸并判断元胞数组 A 的数据类型。

```
A =
    [2x2x2 double]    'Hello'                    [  3.1416   ]
    [17            ]    [1.0000+ 1.0000i]        [5x5 double]
>> D=cellfun('size',A,1)
D =
     2     1     1
     1     1     5
>> E=cellfun('size',A,2)
E =
     2     5     1
     1     1     5
>> F=cellfun('isclass',A,'double')
F =
     1     0     1
     1     1     1
```

❑ **celldisp 函数** 显示所有元胞数组的内容。

【例 2-59】 使用 celldisp 函数显示元胞数组 A 的内容。

```
>> A={rand(2,2,2),'Hello',pi;17,1+i,magic(5)}
A =
    [2x2x2 double]    'Hello'                [      3.1416]
    [            17]    [1.0000+ 1.0000i]    [5x5 double]
>> celldisp(A)
A{1,1} =

(:,:,1) =
    0.1389    0.1987
    0.2028    0.6038
(:,:,2) =

    0.2722    0.0153
    0.1988    0.7468
A{2,1} =
```

```
    17

A{1,2} =
     Hello

A{2,2} =
   1.0000 + 1.0000i

A{1,3} =
    3.1416
A{2,3} =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
```

❑ **cellplot 函数** 利用图形方式显示元胞数组。

【例 2-60】 使用 cellplot 函数显示元胞数组 A。

cellplot 函数实例如图 2-3 所示。

```
>> A={rand(2,2,2),'Hello',pi;17,1+i,magic(5)}
A =
    [2x2x2 double]    'Hello'              [      3.1416]
    [            17]    [1.0000+ 1.0000i]    [5x5 double]
>> cellplot(A)
```

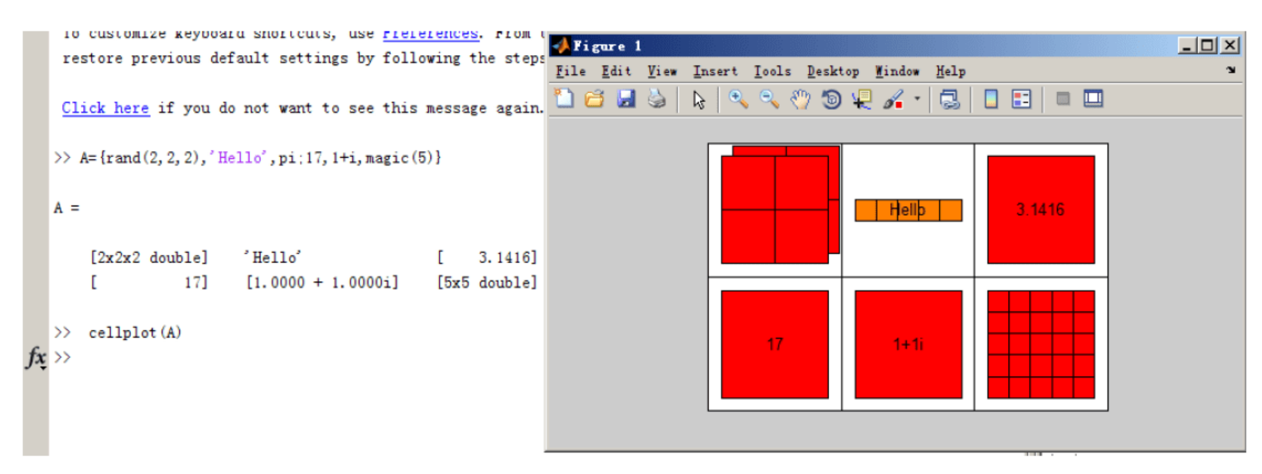


图 2-3 cellplot 函数实例

❑ **cell2mat 函数** 将元胞数组转变成为普通的矩阵。

【例 2-61】 cell2mat 函数样例。

```
>> A = {[1] [2 3 4]; [5; 9] [6 7 8; 10 11 12]}
A =
    [1]                  [1x3 double]
    [2x1 double]    [2x3 double]
>>  B = cell2mat(A)
B =
     1      2      3      4
```




```
     5     6     7     8
     9    10    11    12
>> a={[1 2 3;5 6 7],[4;8];[9 10 ],[11 12]}
a =
    [2x3 double]    [2x1 double]
    [1x2 double]    [1x2 double]
>> b= cell2mat(a)
b =
     1     2     3     4
     5     6     7     8
     9    10    11    12
>> C={[1 2;5 6],[3 4];[9 10],[7 8;11 12]}
C =
    [2x2 double]    [1x2 double]
    [1x2 double]    [2x2 double]
>> D= cell2mat(C)
??? Error using ==> cat
CAT arguments dimensions are not consistent.
Error in ==> C:\MATLAB6p5p1\toolbox\MATLAB\datatypes\cell2mat.m
On line 85  ==>         m{n} = cat(2,c{n,:});
```

❑ **mat2cell 函数**　将普通的矩阵转变为元胞数组。

【例 2-62】 使用 mat2cell 函数将矩阵 X 转变为元胞数组。

```
>> X = [1 2 3 4; 5 6 7 8; 9 10 11 12]
X =
     1     2     3     4
     5     6     7     8
     9    10    11    12
>> Y = mat2cell(X,[1 2],[1 3])
Y =
    [  1]           [1x3 double]
    [2x1 double]    [2x3 double]
```

❑ **num2cell 函数**　将数值数组转变为元胞数组。

【例 2-63】 使用 num2cell 函数将数值数组 X 转变为元胞数组。

```
>> X = [1 2 3 4; 5 6 7 8; 9 10 11 12]
X =
     1     2     3     4
     5     6     7     8
     9    10    11    12
>> Y = num2cell(X)
Y =
    [1]    [ 2]    [ 3]    [ 4]
    [5]    [ 6]    [ 7]    [ 8]
    [9]    [10]    [11]    [12]

>> Y = num2cell(X,2)
Y =
    [1x4 double]
```

```
    [1x4 double]
    [1x4 double]
>> Z = num2cell(X,1)
Z =
    [3x1 double]    [3x1 double]    [3x1 double]    [3x1 double]
>> M = num2cell(X,[1,2])
M =
    [3x4 double]
1     2     3     4
5     6     7     8
9    10    11    12

1     2     3     4
5     6     7     8
9    10    11    12

1     2     3     4
5     6     7     8
9    10    11    12
```

2.2 数组及其函数

所谓数组，就是相同数据类型的元素按一定顺序排列的集合，就是把有限个类型相同的变量用一个名字命名，然后用编号区分它们的变量的集合，这个名字成为数组名，编号成为下标。组成数组的各变量成为数组的分量，也称为数组的元素，有时也称为下标变量。数组是在程序设计中，为了处理方便， 把具有相同类型的若干变量按有序的形式组织起来的一种形式，这些按序排列的同类数据元素的集合称为数组。

MATLAB 的一个重要功能就是能够进行向量和矩阵运算，因此，向量和矩阵 MATLAB 具有非常重要作用。MATLAB 中向量和矩阵主要用数组来表示，数组是 MATLAB 的核心数据结构。

2.2.1 数组的建立

数组的创建包括一维数组和二维数组的创建。一维数组的创建包括一维行向量和一维列向量的创建。创建一维行向量和一维列向量的主要区别在于创建数组时，数组元素是行排列还是列排列。

创建一维行向量即以左方括号开始，以空格或逗号为间隔输入元素值，最后以右方括号结束。由于数组元素值以空格隔开，复数作为数组元素时中间不能键入空格。

MATLAB 中可以利用冒号生成等差数组。语法为数组名=起始值：增量：结束值。增量为正，代表递增，增量为负，代表递减，默认增量为 1。

创建一维列向量，则需要把所有数组元素用分号隔开，并用方括号把数组元素括起来，也可通过转置运算符’将已经创建好的行向量转置为列向量。

创建二维数组与创建一维数组的方式类似。在创建二维数组时，用逗号或空格区分同一行的不同元素，用分号或回车区分不同行。

【例 2-64】 创建二维数组。

```
>> A=[1,2,3,4,5,6,7,8,9]
A =
     1     2     3     4     5     6     7     8     9
>> A=1:9
A =
     1     2     3     4     5     6     7     8     9
>> A=1:2:9
A =
     1     3     5     7     9
>>A=[1;2;3;4;5;6]
A =
     1
     2
     3
     4
     5
     6
>> A=[1 1+i 2-i 3 5];
>> B=A'
B =
   1.0000
   1.0000 - 1.0000i
   2.0000 + 1.0000i
   3.0000
   5.0000
>> A=[1,2,3;4,5,6]
A =
     1     2     3
     4     5     6
```

MATLAB 还提供了大量的库函数用于生成特殊的数组，如表 2-15 所示。

表 2-15　生成特殊数组的函数

函　数	功　能	语　法	备　注
eye	生成单位矩阵	Y=eye(n) Y=eye(m,n) Y= eye(size(A))	
linspace	生成线性分布的向量	y=linspace(a,b) y= linspace(a,b,n)	生成从 a 到 b 之间的 n 个（默认值 100）均匀数
ones	用于生成全部元素为 1 的数组	Y = ones(n) Y=ones(m,n) Y=ones([m n]) Y= ones(size(A))	

续表

函　数	功　能	语　法	备　注
rand	生成随机数组，数组元素值均匀分布	Y=rand Y=rand(n) Y=rand(m,n) Y=rand(size(A))	
randn	生成随机数组，数组元素值正态分布	Y=randn Y=randn(n) Y=randn(m,n) Y=randn(size(A))	
zeros	用于生成全部元素为 0 的数组	Y=zeros(n) Y=zeros(m,n) Y=zeros(size(A))	

2.2.2　数组的操作

1．数组寻址

数组中包含多个元素，因此，对数组的单个元素或多个元素进行访问操作时，需要对数组进行寻址操作。在 MATLAB 中，数组寻址通过对数组下标的访问实现，MATLAB 中提供 end 参数表示数组的末尾。

MATLAB 在内存中以列的方向保存二维数组，对于一个 m 行 n 列的数组，i、j 分别表示行、列的索引，二维数组的寻址可表示为 A(i,j)；如果采用单下标寻址，则数组中元素的下标 k 表示为（j-1）*m+i。

【例 2-65】 数组寻址样例。

```
>> A=randn(1,6)
A =
    0.8156    0.7119    1.2902    0.6686    1.1908   -1.2025
>> A(5)
ans =
    1.1908
>> A([1 3 4 6])
ans =
    0.8156    1.2902    0.6686   -1.2025
>> A(3:5)
ans =
    1.2902    0.6686    1.1908
>> A(3:end)
ans =
    1.2902    0.6686    1.1908   -1.2025

>> A=randn(3,4)
A =
   -0.0198    0.2573   -0.8051   -0.9219
   -0.1567   -1.0565    0.5287   -2.1707
   -1.6041    1.4151    0.2193   -0.0592
```

```
>> A(6)
ans =
   1.4151
>> A(3,2)
ans =
   1.4151
```

2. 数组的扩展与裁剪

数组的扩展指改变数组现有的大小，增加新的数组元素，使得数组的行数或列数增加；而数组的裁剪指从现有的数组中抽出部分数组元素，组成一个维数更小的新数组。

（1）数组的扩展

赋值扩展是数组扩展中较为常用的方法。如果有一个 m 行 n 列的数组 A，要通过赋值来扩展该数组，可以使用超出目前数组大小的索引数字，并对该位置的数组元素进行赋值来完成对数组的扩展，同时未指定的新位置默认赋值为 0。

【例 2-66】 将数组的扩展样例。

```
>> X=[1 2 3;4 5 6;7 8 9];            %数组的赋值扩展
>> X(4,4)=10
X =
     1     2     3     0
     4     5     6     0
     7     8     9     0
     0     0     0    10
>> X(:,5)=20
X =
     1     2     3     0    20
     4     5     6     0    20
     7     8     9     0    20
     0     0     0    10    20
>>  xx=X(:,[1:5,1:5])                   %多次寻址扩展
xx =
     1     2     3     0    20     1     2     3     0    20
     4     5     6     0    20     4     5     6     0    20
     7     8     9     0    20     7     8     9     0    20
     0     0     0    10    20     0     0     0    10    20
>> Y=ones(2,5)
Y =
     1     1     1     1     1
     1     1     1     1     1
>> xy_r=[X;Y]                           %列合成扩展
xy_r =
     1     2     3     0    20
     4     5     6     0    20
     7     8     9     0    20
     0     0     0    10    20
     1     1     1     1     1
     1     1     1     1     1
>> xy_c=[X,Y(:,1:4)']                  %行合成扩展
xy_c =
```

```
      1    2    3    0   20    1    1
      4    5    6    0   20    1    1
      7    8    9    0   20    1    1
      0    0    0   10   20    1    1
```

（2）数组的裁剪

MATLAB 中通常采用冒号操作符裁剪数组，冒号操作符的使用方法为

```
B=A([x1,x2,…],[y1,y2,…])
```

其中，[x1,x2,...]表示行索引向量，[y1,y2,...]表示列索引向量，该式表示提取数组 A 的 x1,x2,...等行，y1,y2,...等列，组成一个新的数组。当某一索引值的位置上不是数字，而是冒号，则表示提取此索引位置的所有数组元素。

3．数组元素的删除

删除数组元素，可以通过将该位置的数组元素赋值为空方括号[]，一般配合冒号使用，将数组中的某些行、列元素删除。注意，在进行数组元素的删除时，索引值必须是完整的行或列，而不能是数组内部的元素块或单个元素。

【例 2-67】 数组元素的删除样例。

```
>> X=rand(6,6)
X =
    0.9501    0.4565    0.9218    0.4103    0.1389    0.0153
    0.2311    0.0185    0.7382    0.8936    0.2028    0.7468
    0.6068    0.8214    0.1763    0.0579    0.1987    0.4451
    0.4860    0.4447    0.4057    0.3529    0.6038    0.9318
    0.8913    0.6154    0.9355    0.8132    0.2722    0.4660
    0.7621    0.7919    0.9169    0.0099    0.1988    0.4186
>> X(3,:)
ans =
    0.6068    0.8214    0.1763    0.0579    0.1987    0.4451
>> X(1:2:6,2:2:6)                    %圆括号中表示为初值、步长、终值
ans =
    0.4565    0.4103    0.0153
    0.8214    0.0579    0.4451
    0.6154    0.8132    0.4660
>> X([1,2,5],[2,3,6])                   %方括号中表示取值
ans =
    0.4565    0.9218    0.0153
    0.0185    0.7382    0.7468
    0.6154    0.9355    0.4660
>> X([1,2],:)=[]
X =
    0.6068    0.8214    0.1763    0.0579    0.1987    0.4451
    0.4860    0.4447    0.4057    0.3529    0.6038    0.9318
    0.8913    0.6154    0.9355    0.8132    0.2722    0.4660
    0.7621    0.7919    0.9169    0.0099    0.1988    0.4186
>> X(1:2:4,:)=[]
X =
    0.4860    0.4447    0.4057    0.3529    0.6038    0.9318
```

```
    0.7621    0.7919    0.9169    0.0099    0.1988    0.4186
>> X(:,[1,2,3,4])=[]
X =
    0.6038    0.9318
    0.1988    0.4186
```

4．数组的查找和排序

1）数组的查找

MATLAB 提供数组查找函数 find，它能够查找数组中的非零数组元素，并返回其数组索引值。find 函数的语法为

- indices = find(X)
- indices = find(X, k)
- indices = find(X, k, 'first')
- indices = find(X, k, 'last')
- [i,j] = find(...)
- [i,j,v] = find(...)

其中，indices 表示非零元素的下标值，i，j 分别表示行下标向量和列下标向量，v 表示非零元素向量。

在 MATLAB 的实际应用中，经常通过多重逻辑关系组合产生逻辑数组，判断数组元素是否满足某种比较关系，然后通过 find 函数返回符合比较关系的元素索引，从而实现数组元素的查找。

2）数组的排序

MATLAB 提供数组排序函数 sort，该函数可对任意给定的数组进行排序。sort 函数的语法为

- B = sort(A)
- B = sort(A,dim)
- B = sort(...,mode)
- [B,IX] = sort(...)

其中，B 为返回的排序后的数组，A 为输入待排序数组，当 A 为多维数组时，用 dim 指定需要排序的维数（默认为 1）；mode 为排序的方式，可以取值为 ascend 和 descend，分别表示升序和降序，默认为升序；IX 用于存储排序后的下标数组。

【例 2-68】 数组的查找和排序样例。

```
>> X = [3 2 0; -5 0 7; 0 0 1]
>> [i,j]=find((X>2)&(X<9))
i =
     1
     2
j =
     1
     3
>> sort(X,1)   %以列维方向排序
ans =
```

```
    -5     0     0
     0     0     1
     3     2     7
>> sort(X,1,'descend')     %'1'表示列维降序排序
ans =
     3     2     7
     0     0     1
    -5     0     0
>>  [B,IX] = sort(X,2)       %'2'表示列维降序排序
B =
     0     2     3
    -5     0     7
     0     0     1
IX =
     3     2     1
     1     2     3
     1     2     3
```

5. 数组的运算

MATLAB 中数组的加减乘除运算是按元素对元素方式进行的。数组的加减法为数组对应元素的加减法，利用运算符“+”和“−”实现该运算。相加或相减的两个数组必须有相同的维数，或者是数组同标量相加减。

数组的乘除法为对应数组元素的乘除，通过运算符“.*”和“./”实现。相乘或相除的两个数组必须具有相同的维数，或者是数组同标量相加减。

数组幂运算用符号“.^”实现，表示对元素的幂。数组幂运算以三种方式进行：底为数组、底为标量和底与指数均为数组。当底和指数均为数组时，要求两个数组具有相同的维数。

【例 2-69】 数组的加减乘除等运算样例。

```
>> A=ones(3,3)
A =
     1     1     1
     1     1     1
     1     1     1
>> B=rand(3)
B =
    0.8462    0.6721    0.6813
    0.5252    0.8381    0.3795
    0.2026    0.0196    0.8318
>> C1=A+B
C1 =
    1.8462    1.6721    1.6813
    1.5252    1.8381    1.3795
    1.2026    1.0196    1.8318
>> C2=A-B
C2 =
    0.1538    0.3279    0.3187
    0.4748    0.1619    0.6205
```

```
    0.7974    0.9804    0.1682
>> C3=A.*B
C3 =
    0.8462    0.6721    0.6813
    0.5252    0.8381    0.3795
    0.2026    0.0196    0.8318
>> C4=A./B
C4 =
    1.1817    1.4878     1.4678
    1.9042    1.1931     2.6352
    4.9347   50.9178     1.2022
>> A=[1 2 3 4;5 6 7 8;9 10 11 12];
>> B=[1 1 1 1;2 2 2 2;3 3 3 3];
>> A.^2
ans =
     1         4         9        16
    25        36        49        64
    81       100       121       144
>> 2.^A
ans =
     2         4         8        16
    32        64       128       256
   512      1024      2048      4096
>> A.^B
ans =
     1         2         3         4
    25        36        49        64
   729      1000      1331      1728
```

6. 数组操作函数

MATLAB 中提供了大量库函数对数组进行特定的操作，如表 2-16 所示。

表 2-16　对数组进行特定操作的库函数

函　数	语　法	说　明
cat	C=cat(dim, A, B)	按指定维方向，扩展数组
diag	X=diag(v,k)X=diag(v)v=diag(X,k)v=diag(X)	提取对角元素或生成对角矩阵。k=0 表示主对角线，k>0 表示对角线上方，k<0 表示对角线下方
flipud	B=flipud(A)	以数组水平中线为对称轴，交换上下对称位置上的数组元素
fliplr	B=fliplr(A)	以数组垂直中线为对称轴，交换左右对称位置上的数组元素
repmat	B=repmat(A,m,n)	以指定的行数和列数复制数组 A
reshape	B=reshape(A,m,n)	以指定的行数和列数重新排列数组 A
size	[m,n]=size(X)m=size(X,dim)	返回数组的行数和列数
length	n=length(X)	返回 max(size(x))

【例 2-70】 操作数组函数样例。

```
>> A=[1,2;3,4];
>> B=[5,6;7,8];
```

```
>> cat(1,A,B)
ans =
     1     2
     3     4
     5     6
     7     8
>> cat(2,A,B)
ans =
     1     2     5     6
     3     4     7     8
>> A=rand(5)
A =
    0.0592    0.8744    0.7889    0.3200    0.2679
    0.6029    0.0150    0.4387    0.9601    0.4399
    0.0503    0.7680    0.4983    0.7266    0.9334
    0.4154    0.9708    0.2140    0.4120    0.6833
    0.3050    0.9901    0.6435    0.7446    0.2126
>> x=diag(A,1)
x =
    0.8744
    0.4387
    0.7266
    0.6833
>> B=diag(x,1)
B =
         0    0.8744         0         0         0
         0         0    0.4387         0         0
         0         0         0    0.7266         0
         0         0         0         0    0.6833
         0         0         0         0         0
A =
    0.0592    0.8744    0.7889    0.3200    0.2679
    0.6029    0.0150    0.4387    0.9601    0.4399
    0.0503    0.7680    0.4983    0.7266    0.9334
    0.4154    0.9708    0.2140    0.4120    0.6833
    0.3050    0.9901    0.6435    0.7446    0.2126
>> C=flipud(B)
C =
         0         0         0         0         0
         0         0         0         0    0.6833
         0         0         0    0.7266         0
         0         0    0.4387         0         0
         0    0.8744         0         0         0
>> D=fliplr(B)
D =
         0         0         0    0.8744         0
         0         0    0.4387         0         0
         0    0.7266         0         0         0
    0.6833         0         0         0         0
         0         0         0         0         0
>> A=randn(2)
```

```
A =
   -0.4326    0.1253
   -1.6656    0.2877
>> B=repmat(A,1,2)
B =
   -0.4326    0.1253   -0.4326    0.1253
   -1.6656    0.2877   -1.6656    0.2877
>> C=reshape(B,4,2)
C =
   -0.4326   -0.4326
   -1.6656   -1.6656
    0.1253    0.1253
    0.2877    0.2877
>> size(C)
ans =
     4     2
>> size(C,1)
ans =
     4
```

2.3 矩阵及其函数

矩阵（Matrix）是指纵横排列的二维数据表格，最早来自于方程组的系数及常数所构成的方阵。矩阵的研究历史悠久，拉丁方阵和幻方在史前年代已有人研究。在数学名词中，矩阵用来表示统计数据等方面的各种有关联的数据，这个定义很好地解释了 Matrix 代码制造世界的数学逻辑基础。

阵的运算是数值分析领域的重要问题。将矩阵分解为简单矩阵的组合可以在理论和实际应用上简化矩阵的运算。对一些应用广泛而形式特殊的矩阵，例如，稀疏矩阵和准对角矩阵，有特定的快速运算算法。在天体物理、量子力学等领域，也会出现无穷维的矩阵，是矩阵的一种推广。

MATLAB 意为矩阵工厂（矩阵实验室），MATLAB 的基本数据单位就是矩阵，它的指令表达式与数学、工程中常用的形式相似，可见学好 MATLAB 的矩阵运算是及其重要且非常具有实用价值的。

有些读者常常将二维数组和矩阵相互混淆，其他书中对于此说明也有所疏忽，故笔者在这里谈下二维数组和矩阵的关系：二维数组具有线性变换含义时，称为矩阵，否则，称为数组。从数据结构的形式上，两者没有区别。

2.3.1 矩阵的建立

在 MATLAB 中，有多种矩阵的创建方法，用户在使用时应根据实际情况，选择最优方法。

（1）直接输入法。

将矩阵的元素用方括号括起来，按矩阵行的顺序输入各元素，同一行的各元素之间用

空格或逗号分隔，不同行的元素之间用分号分隔。

【例 2-71】 直接输入法建立矩阵 A。

```
>>A = [16  3  2  13;5  10  11  8;9  6  7  12;4  15  14  1]
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
```

（2）M 文件建立矩阵。

对于比较大且比较复杂的矩阵，可以为它专门建立一个 M 文件。

具体方法是启动有关编辑程序或 MATLAB 文本编辑器，并输入待建矩阵。把输入的内容存盘（设文件名为 mymatrix.m）。运行该 M 文件，就会自动建立一个名为 A 的矩阵，可供以后使用。

（3）利用矩阵编辑器 Array Editor 创建矩阵。

先在命令窗口输入

```
>>A=1
```

再在 Workspace 窗口中双击该变量，打开矩阵编辑器，进行输入和修改。

（4）特殊矩阵的建立，如表 2-19 所示。

表 2-19　特殊矩阵的建立函数

函　　数	说　　明
zeros	产生元素全为 0 的矩阵
ones	产生元素全为 1 的矩阵
eye	产生单位矩阵
rand	产生均匀分布的随机数矩阵，数值范围（0，1）
randn	产生均值为 0，方差为 1 的正态分布随机数矩阵
diag	获取矩阵的对角线元素，也可生成对角矩阵
tril	产生下三角矩阵
triu	产生上三角矩阵
pascal	产生帕斯卡矩阵
magic	产生魔方阵
vander	产生以向量 V 为基础向量的范得蒙矩阵
hilb	产生希尔伯特矩阵
toeplitz	产生托普利兹矩阵
compan	产生伴随矩阵

❑ **Zeros**　产生全 0 矩阵（零矩阵）。

【例 2-72】 建立一个 3×3 零矩阵。

```
>> zeros(3)
ans =
     0     0     0
     0     0     0
```



```
     0     0     0
```

【例 2-73】 建立一个 3×2 零矩阵。

```
>> zeros(3,2)
ans =
     0     0
     0     0
     0     0
```

【例 2-74】设 A 为 2×3 矩阵，则可以用 zeros(size(A))建立一个与矩阵 A 同样大小的零矩阵。

```
>> A=[1 2 3;4 5 6];     %产生一个 2×3 阶矩阵 A
>> zeros(size(A))       %产生一个与矩阵 A 同样大小的零矩阵
ans =
     0     0     0
     0     0     0
```

- ❑ **ones** 产生全 1 矩阵（幺矩阵）。
- ❑ **eye** 产生单位矩阵。
- ❑ **rand** 产生 0～1 间均匀分布的随机矩阵。

【例 2-75】 创建在区间[20,50]内均匀分布的 5 阶随机矩阵。

```
>> x=20+(50-20)*rand(5)
x =
   44.4417   22.9262   24.7284   24.2566   39.6722
   47.1738   28.3549   49.1178   32.6528   21.0714
   23.8096   36.4064   48.7150   47.4721   45.4739
   47.4013   48.7252   34.5613   43.7662   48.0198
   38.9708   48.9467   44.0084   48.7848   40.3621
```

【例 2-76】 创建均值为 0.6，方差为 0.1 的 5 阶正态分布随机矩阵。

```
>> y=0.6+sqrt(0.1)*randn(5)
y =
    0.9272    0.8809    1.0549    0.5677    0.5905
    0.8299    0.2373    0.7028    0.5236    0.5479
    0.5040    0.2620    0.3613    0.7009    0.7985
    0.6929    0.3440    1.0333    0.6989    0.9457
    0.3510   -0.3311    0.0588    0.3265    0.9508
```

此外，常用的函数还有 reshape(A,m,n)，它在矩阵总元素保持不变的前提下，将矩阵 A 重新排成 m×n 的二维矩阵。

- ❑ **diag** 获取矩阵的对角线元素，也可生成对角矩阵。

【例 2-77】 使用 diag 函数获取矩阵 A 的对角线元素。

```
>> A = [16  3  2  13;5  10  11  8;9  6  7  12;4  15  14  1]
A =
    16     3     2    13
```

```
     5    10    11     8
     9     6     7    12
     4    15    14     1
>> diag(A)
ans =
    16
    10
     7
     1
```

❑ **tril** 产生下三角矩阵。

【例 2-78】 使用 tril 函数将矩阵 A 生成下三角矩阵。

```
>> A = [16  3  2  13;5  10  11  8;9  6  7  12;4  15  14  1]
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
>> tril(A)
ans =
    16     0     0     0
     5    10     0     0
     9     6     7     0
     4    15    14     1
```

❑ **triu** 产生上三角矩阵。

【例 2-79】 使用 triu 函数将矩阵 A 生成上三角矩阵。

```
>> A = [16  3  2  13;5  10  11  8;9  6  7  12;4  15  14  1]
A =
    16     3     2    13
     5    10    11     8
     9     6     7    12
     4    15    14     1
>> triu(A)
ans =
    16     3     2    13
     0    10    11     8
     0     0     7    12
     0     0     0     1
```

❑ **pascal** 产生帕斯卡矩阵。

帕斯卡矩阵是由杨辉三角形表组成的矩阵，杨辉三角形表是二次项(x+y)^n 展开后的系数随自然数 n 的增大组成的一个三角形表。

【例 2-80】 生成一个 5 阶帕斯卡阵。

```
>> P=pascal(5)
P =
         1             1             1             1             1
```




```
        1            2            3            4            5
        1            3            6           10           15
        1            4           10           20           35
        1            5           15           35           70
```

【例 2-81】 求(x+y)5 的展开式。

```
>> pascal(6)
ans =
  Columns 1 through 5
        1            1            1            1            1
        1            2            3            4            5
        1            3            6           10           15
        1            4           10           20           35
        1            5           15           35           70
        1            6           21           56          126
  Column 6  %矩阵次对角线上的元素 1,5,10,10,5,1 即为展开式的系数。
        1
        6
       21
       56
      126
      252
```

❑ **magic** 产生魔方矩阵。

魔方矩阵有一个有趣的性质，其每行、每列及两条对角线上的元素和都相等。对于 n 阶魔方矩阵，其元素由 1，2，3，…，n^2 共 n^2 个整数组成。MATLAB 提供了求魔方矩阵的函数 magic(n)，其功能是生成一个 n 阶魔方矩阵。

【例 2-82】 使用 magic 函数产生魔方矩阵。

```
>> A=magic(6)
A =
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11
```

【例 2-83】 将 101～125 等 25 个数填入一个 5 行 5 列的表格中，使其每行每列及对角线的和均为 565。

```
>> M=100+magic(5)
M =
   117   124   101   108   115
   123   105   107   114   116
   104   106   113   120   122
   110   112   119   121   103
   111   118   125   102   109
```

❑ **vander(V)** 生成以向量 V 为基础向量的范得蒙矩阵。

范得蒙（Vandermonde）矩阵最后一列全为 1，倒数第二列为一个指定的向量，其他各列是其后列与倒数第二列的点乘积，可以用一个指定向量生成一个范得蒙矩阵。

【例 2-84】 生成范得蒙矩阵。

```
>> A=vander([3;4;3;5])
A =
    27     9     3     1
    64    16     4     1
    27     9     3     1
   125    25     5     1
```

❑ **hilb(n)** 生成希尔伯特矩阵。

希尔伯特矩阵（Hilbert matrix）是一种数学变换矩阵，正定且高度病态（即任何一个元素发生一点变动，整个矩阵的值和逆矩阵都会发生巨大变化），病态程度和阶数相关。故使用一般方法求逆会因为原始数据的微小扰动而产生不可靠的计算结果。MATLAB 中，有一个专门求希尔伯特矩阵的逆的函数 invhilb(n)，其功能是求 n 阶的希尔伯特矩阵的逆矩阵。

【例 2-85】 求 5 阶希尔伯特矩阵及其逆矩阵。

```
>> format rat     %以有理形式输出
>> H=hilb(5)
H =
     1             1/2           1/3           1/4           1/5
     1/2           1/3           1/4           1/5           1/6
     1/3           1/4           1/5           1/6           1/7
     1/4           1/5           1/6           1/7           1/8
     1/5           1/6           1/7           1/8           1/9
>> H=invhilb(5)
H =
      25          -300          1050         -1400           630
    -300          4800        -18900         26880        -12600
    1050        -18900         79380       -117600         56700
   -1400         26880       -117600        179200        -88200
     630        -12600         56700        -88200         44100
```

❑ **toeplitz** 生成托普利兹矩阵。

托普利兹（Toeplitz）矩阵除第一行第一列外，其他每个元素都与左上角的元素相同，即主对角线上的元素相等，平行于主对角线的线上的元素也相等。生成托普利兹矩阵的函数是 toeplitz(x,y)，它生成一个以 x 为第一列，y 为第一行的托普利兹矩阵。这里 x, y 均为向量，两者不必等长。toeplitz(x)用向量 x 生成一个对称的托普利兹矩阵。

【例 2-86】 生成一个托普利兹矩阵。

```
>> T=toeplitz(1:6)
T =
  Columns 1 through 5
      1           2           3           4           5
```

```
     2          1          2          3          4
     3          2          1          2          3
     4          3          2          1          2
     5          4          3          2          1
     6          5          4          3          2
  Column 6
     6
     5
     4
     3
     2
     1
```

- **compan** 生成伴随矩阵。

MATLAB 生成伴随矩阵的函数是 compan(p)，其中，p 是一个多项式的系数向量，高次幂系数排在前，低次幂排在后。

【例 2-87】 求多项式的 x^3-7x+6 的伴随矩阵。

```
>> p=[1,0,-7,6];
>> compan(p)
ans =
     0          7          -6
     1          0          0
     0          1          0
```

（5）行向量的建立

- **冒号表达式建立向量** 冒号是一个重要的运算符。形式为 e1:e2:e3，其中，e1 为初始值，e2 为步长，e3 为终止值。以 e1 为开始，e3 为结束，步长为 e2。
- **Linspace 建立行向量** 形式 linspace(a，b，n)，a 和 b 分别为行向量的第一个和最后一个元素，n 为总元素，省略 n 自动产生 100 个元素的行向量。

2.3.2 矩阵运算

1. 算术运算

MATLAB 的基本算术运算有＋（加）、－（减）、*（乘）、/（右除）、\（左除）、^（乘方）、'（转置）。运算是在矩阵意义下进行的，单个数据的算术运算只是一种特例。

1）矩阵加减运算

假定有两个矩阵 A 和 B，则可以由 A+B 和 A–B 实现矩阵的加减运算。运算规则是若 A 和 B 矩阵的维数相同，则可以执行矩阵的加减运算，A 和 B 矩阵的相应元素相加减。如果 A 与 B 的维数不相同，则 MATLAB 将给出错误信息，提示用户两个矩阵的维数不匹配。

2）矩阵乘法运算

假定有两个矩阵 A 和 B，若 A 为 m*n 矩阵，B 为 n*p 矩阵，则 C=A*B 为 m*p 矩阵。

3）矩阵除法运算

在 MATLAB 中，有两种矩阵除法运算：\和/，分别表示左除和右除。如果 A 矩阵是非奇异方阵，则 A\B 和 B/A 运算可以实现。A\B 等效于 A 的逆左乘 B 矩阵，也就是 inv(A) *B，而 B/A 等效于 A 矩阵的逆右乘 B 矩阵，也就是 B*inv(A)。对于含有标量的运算，两种除法运算的结果相同。对于矩阵来说，左除和右除表示两种不同的除数矩阵和被除数矩阵的关系，一般 A\B≠B/A。

4）矩阵的乘方运算

一个矩阵的乘方运算可以表示成 A^x，要求 A 为方阵，x 为标量。

5）矩阵的转置运算

对实数矩阵进行行列互换，对复数矩阵，共轭转置，特殊的操作符.'共轭不转置（见点运算）。

6）点运算

在 MATLAB 中，有一种特殊的运算，因为其运算符是在有关算术运算符前面加点，所以叫点运算。点运算符有. *、./、.\和.^。两矩阵进行点运算是指它们的对应元素进行相关运算，要求两矩阵的维参数相同。

2. 关系运算

MATLAB 提供了 6 种关系运算符：<（小于）、<=（小于或等于）、>（大于）、>=（大于或等于）、==（等于）、~=（不等于）。关系运算符的运算法则如下。

（1）当两个比较量是标量时，直接比较两数的大小。若关系成立，关系表达式结果为 1，否则为 0。

（2）当参与比较的量是两个维数相同的矩阵时，比较是对两矩阵相同位置的元素按标量关系运算规则逐个进行，并给出元素比较结果。最终的关系运算的结果是一个维数与原矩阵相同的矩阵，其元素由 0 或 1 组成。

（3）当参与比较的一个是标量，而另一个是矩阵时，则把标量与矩阵的每一个元素按标量关系运算规则逐个比较，并给出元素比较结果。最终的关系运算的结果是一个维数与原矩阵相同的矩阵，其元素由 0 或 1 组成。

3. 逻辑运算

MATLAB 提供了 3 种逻辑运算符：&（与）、|（或）和~（非）。逻辑运算的运算法则如下。

（1）在逻辑运算中，确认非零元素为真，用 1 表示，零元素为假，用 0 表示。

（2）设参与逻辑运算的是两个标量 a 和 b，那么，a&b a,b 全为非零时，运算结果为 1，否则为 0。 a|b a,b 中只要有一个非零，运算结果为 1。~a 当 a 是零时，运算结果为 1；当 a 非零时，运算结果为 0。

（3）若参与逻辑运算的是两个同维矩阵，那么，运算将对矩阵相同位置上的元素按标量规则逐个进行。最终运算结果是一个与原矩阵同维的矩阵，其元素由 1 或 0 组成。

（4）若参与逻辑运算的一个是标量，另一个是矩阵，那么运算将在标量与矩阵中的每个元素之间按标量规则逐个进行。最终运算结果是一个与矩阵同维的矩阵，其元素由 1 或 0 组成。

（5）逻辑非是单目运算符，也服从矩阵运算规则。

（6）在算术、关系、逻辑运算中，算术运算优先级最高，逻辑运算优先级最低。

2.3.3 矩阵分析

1. 对角阵与三角阵

1）对角阵

只有对角线上有非 0 元素的矩阵称为对角矩阵，对角线上的元素相等的对角矩阵称为数量矩阵，对角线上的元素都为 1 的对角矩阵称为单位矩阵。

❑ **提取矩阵的对角线元素**

设 A 为 m×n 矩阵，diag(A)函数用于提取矩阵 A 主对角线元素，产生一个具有 min(m,n)个元素的列向量。diag(A)函数还有一种形式 diag(A,k)，其功能是提取第 k 条对角线的元素。

❑ **构造对角矩阵**

设 V 为具有 m 个元素的向量，diag(V)将产生一个 m×m 对角矩阵，其主对角线元素即为向量 V 的元素。diag(V)函数也有另一种形式 diag(V,k)，其功能是产生一个 n×n(n=m+k)对角阵，其第 m 条对角线的元素即为向量 V 的元素。

【例 2-88】 先建立 5×5 矩阵 A，然后将 A 的第一行元素乘以 1，第二行乘以 2，…，第五行乘以 5。

```
>> A=[17,0,1,0,15;23,5,7,14,16;4,0,13,0,22;10,12,19,21,3;11,18,25,2,19];
>> D=diag(1:5);
>> D*A                %用 D 左乘 A，对 A 的每行乘以一个指定常数
ans =
     17          0          1          0         15
     46         10         14         28         32
     12          0         39          0         66
     40         48         76         84         12
     55         90        125         10         95
```

2）三角阵

三角阵又进一步分为上三角阵和下三角阵，所谓上三角阵，即矩阵的对角线以下的元素全为 0 的一种矩阵，而下三角阵则是对角线以上的元素全为 0 的一种矩阵。

❑ **上三角矩阵**

求矩阵 A 的上三角阵的 MATLAB 函数是 triu(A)。triu(A)函数也有另一种形式 triu(A,k)，其功能是求矩阵 A 的第 k 条对角线以上的元素。例如，提取矩阵 A 的第 2 条对角线以上的元素，形成新的矩阵 B。

❑ **下三角矩阵**

在 MATLAB 中，提取矩阵 A 的下三角矩阵的函数是 tril(A)和 tril(A,k)，其用法与提取上三角矩阵的函数 triu(A)和 triu(A,k)完全相同。

2. 矩阵的转置与旋转

（1）矩阵的转置操作使用转置运算符单撇号（'）。

（2）矩阵的旋转操作使用函数 rot90(A,k)，其中，k 表示将矩阵 A 旋转 90º 的 k 倍，当 k 为 1 时可省略。

（3）矩阵实施左右翻转是将原矩阵的第一列和最后一列调换，第二列和倒数第二列调换，…，依次类推。MATLAB 对矩阵 A 实施左右翻转的函数是 fliplr(A)，而对矩阵的上下翻转操作使用函数 flipud(A)。

【例 2-89】 A'是矩阵 A 的转置，B 是将矩阵 A 旋转 180°，C 是 A 左右翻转得到的，D 是 A 上下翻转得到的，求 A'、B、C、D。

```
>> A=[17,0,1,0,15;23,5,7,14,16;4,0,13,0,22;10,12,19,21,3;11,18,25,2,19];
>> A
A =
             17             0             1             0            15
             23             5             7            14            16
              4             0            13             0            22
             10            12            19            21             3
             11            18            25             2            19
>> A'
ans =
             17            23             4            10            11
              0             5             0            12            18
              1             7            13            19            25
              0            14             0            21             2
             15            16            22             3            19
>> B=rot90(A,2)
B =
             19             2            25            18            11
              3            21            19            12            10
             22             0            13             0             4
             16            14             7             5            23
             15             0             1             0            17
>> C=fliplr(A)
C =
             15             0             1             0            17
             16            14             7             5            23
             22             0            13             0             4
              3            21            19            12            10
             19             2            25            18            11
>> D=flipud(A)
D =
             11            18            25             2            19
             10            12            19            21             3
              4             0            13             0            22
             23             5             7            14            16
             17             0             1             0            15
```

3. 矩阵的逆与伪逆

1）矩阵的逆

对于一个方阵 A，如果存在一个与其同阶的方阵 B，使得 AB=BA=I（I 为单位矩阵），则称 B 为 A 的逆矩阵，当然，A 也是 B 的逆矩阵。求方阵 A 的逆矩阵可调用函数 inv(A)。

2）矩阵的伪逆

如果矩阵 A 不是一个方阵，或者 A 是一个非满秩的方阵时，矩阵 A 没有逆矩阵，但可以找到一个与 A 的转置矩阵 A'同型的矩阵 B，使得 ABA=A，BAB=B，此时称矩阵 B 为矩阵 A 的伪逆，也称为广义逆矩阵。在 MATLAB 中，求一个矩阵伪逆的函数是 pinv(A)。

3）用矩阵求逆方法求解线性方程组

在线性方程组 Ax=b 两边各左乘 A-1，有

A-1Ax=A-1b

由于 A-1A=I，故得

x=A-1b

【例 2-90】 用求逆矩阵的方法解线性方程组。

```
>> A=[1,2,3;1,4,9;1,8,27]
A =
      1            2            3
      1            4            9
      1            8           27
>> b=[5,-2,6]'
b =
      5
     -2
      6
>> x=inv(A) *b
x =
     23
    -29/2
     11/3
```

📖 也可以运用左除运算符“\”求解线性代数方程组。

4. 方阵的行列式

行列式在数学中，是由解线性方程组产生的一种算式。行列式的特性可以被概括为一个多次交替线性形式，这个本质使得行列式在欧几里德空间中可以成为描述“体积”的函数。

把一个方阵看作一个行列式，并对其按行列式的规则求值，这个值就称为矩阵所对应的行列式的值。在 MATLAB 中，求方阵 A 所对应的行列式的值的函数是 det(A)。

5. 矩阵的秩与迹

（1）矩阵线性无关的行数与列数称为矩阵的秩。在 MATLAB 中，求矩阵秩的函数是 rank(A)。

（2）矩阵的迹等于矩阵的对角线元素之和，也等于矩阵的特征值之和。在 MATLAB 中，求矩阵的迹的函数是 trace(A)。

6. 向量和矩阵的范数

范数是具有“长度”概念的函数。在线性代数、泛函分析及相关的数学领域，泛函是一个函数，其为矢量空间内的所有矢量赋予非零的正长度或大小。半范数反而可以为非零

的矢量赋予零长度。

举一个简单的例子，在二维的欧氏几何空间 R 就可定义欧氏范数。在这个矢量空间中的元素常常在笛卡儿坐标系统中被画成一个从原点出发的带有箭头的有向线段。每一个矢量的欧氏范数就是有向线段的长度。

其中，定义范数的矢量空间就是赋范矢量空间。同样，其中定义半范数的矢量空间就是赋半范矢量空间。

矩阵或向量的范数用来度量矩阵或向量在某种意义下的长度。范数有多种方法定义，其定义不同，范数值也就不同。

（1）向量的 3 种常用范数及其计算函数。在 MATLAB 中，求向量范数的函数如下。

- **cond(A,1)**　计算 A 的 1 阶范数下的条件数；
- **cond(A)或 cond(A,2)**　计算 A 的 2 阶范数数下的条件数；
- **cond(A,inf)**　计算 A 的无穷阶范数下的条件数。

（2）矩阵的范数及其计算函数。MATLAB 提供了求 3 种矩阵范数的函数，其函数调用格式与求向量的范数的函数完全相同。

（3）矩阵的条件数。在 MATLAB 中，计算矩阵 A 的 3 种条件数的函数如下。

- **cond(A,1)**　计算 A 的 1 阶范数下的条件数；
- **cond(A)或 cond(A,2)**　计算 A 的 2 阶范数数下的条件数；
- **cond(A,inf)**　计算 A 的无穷阶范数下的条件数。

7. 矩阵的特征值与特征向量

数学上，线性变换的特征向量（本征向量）是一个非退化的向量，其方向在该变换下不变。该向量在此变换下缩放的比例称为其特征值（本征值）。一个变换通常可以由其特征值和特征向量完全描述。特征空间是相同特征值的特征向量的集合。

在 MATLAB 中，计算矩阵 A 的特征值和特征向量的函数是 eig(A)，常用的调用格式有以下 3 种。

- **E=eig(A)**　求矩阵 A 的全部特征值，构成向量 E。
- **[V,D]=eig(A)**　求矩阵 A 的全部特征值，构成对角阵 D，并求 A 的特征向量构成 V 的列向量。
- **[V,D]=eig(A,'nobalance')**　与第 2 种格式类似，但第 2 种格式中先对 A 作相似变换后求矩阵 A 的特征值和特征向量，而格式 3 直接求矩阵 A 的特征值和特征向量。

【例 2-91】 用求特征值的方法解方程：$3x^5-7x^4+5x^2+2x-18=0$。

```
>> p=[3,-7,0,5,2,-18]
p =
  Columns 1 through 5
     3             -7              0              5              2
  Column 6
   -18
>> A=compan(p)      %A 的伴随矩阵
A =
   7/3          0           -5/3           -2/3              6
     1          0              0              0              0
     0          1              0              0              0
```

```
    0           0           1           0           0
    0           0           0           1           0
>> x1=eig(A)          %求 A 的特征值
x1 =
   5160/2363
     1       +    1i
     1       -    1i
  -1397/1510    +  670/931i
  -1397/1510    -  670/931i
>> x2=roots(p)          %直接求多项式 p 的零点
x2 =
   5160/2363
     1       +    1i
     1       -    1i
  -1397/1510    +  670/931i
  -1397/1510    -  670/931i
```

8．矩阵的超越函数

在数学领域中，超越函数与代数函数相反，是指那些不满足任何以多项式方程的函数，即函数不满足以变量自身的多项式为系数的多项式方程。换句话说，超越函数就是"超出"代数函数范围的函数，也就是说，函数不能表示为有限次的加、减、乘、除和开方的运算。

1）矩阵平方根

sqrtm(A)用来计算矩阵 A 的平方根。

2）矩阵对数

logm(A)计算矩阵 A 的自然对数，此函数输入参数的条件与输出结果间的关系和函数 sqrtm(A)完全相同。

3）矩阵指数

expm(A)、expm1(A)、expm2(A)、expm3(A)的功能都求矩阵指数 e^A。

原理介绍：

expm()是按照下面的方式来计算的。

[V,D] = EIG(X)

EXPM(X) = V*diag(exp(diag(D)))/V

V：X 的特征向量

D：对应的特征值

4）普通矩阵函数

funm(A,'fun')用来计算直接作用于矩阵 A 的由'fun'指定的超越函数值。当 fun 取 sqrt 时，funm(A, 'sqrt')可以计算矩阵 A 的平方根，与 sqrtm(A)的计算结果相同。

2.3.4 稀疏矩阵及其运算

对于一个 n 阶矩阵，通常需要 n2 的存储空间，当 n 很大时，进行矩阵运算时会占用大量的内存空间和运算时间。在许多实际问题中遇到的大规模矩阵中，通常含有大量 0 元

素，这样的矩阵称为稀疏矩阵。MATLAB 支持稀疏矩阵，只存储矩阵的非零元素。由于不存储那些“0”元素，也不对它们进行操作，从而节省内存空间和计算时间，其计算的复杂性和代价仅仅取决于稀疏矩阵的非零元素的个数，这在矩阵的存储空间和计算时间上都有很大的优点。

矩阵的密度定义为矩阵中非零元素的个数除以矩阵中总的元素个数。对于低密度的矩阵，采用稀疏方式存储是一种很好的选择。

1. 稀疏矩阵的创建

稀疏矩阵的创建方法如下。

1）将完全存储方式转化为稀疏存储方式

函数 A=sparse(S)能将矩阵 S 转化为稀疏存储方式的矩阵 A。当矩阵 S 是稀疏存储方式时，则函数调用相当于 A=S。sparse 函数还有其他一些调用格式，如 sparse(m,n)，生成一个 m*n 的所有元素都是 0 的稀疏矩阵。sparse(u,v,S)：u,v,S 是 3 个等长的向量。S 是要建立的稀疏矩阵的非 0 元素，u(i)、v(i)分别是 S(i)的行和列下标，该函数建立一个 max(u)行、max(v)列并以 S 为稀疏元素的稀疏矩阵。此外，还有一些和稀疏矩阵操作有关的函数。full(A)：返回和稀疏存储矩阵 A 对应的完全存储方式矩阵。

2）直接创建稀疏矩阵

```
S=sparse(i,j,s,m,n)
```

其中，i 和 j 分别是矩阵非零元素的行和列指标向量，s 是非零元素值向量，m，n 分别是矩阵的行数和列数。

3）从文件中创建稀疏矩阵

利用 load 和 spconvert 函数可以从包含一系列下标和非零元素的文本文件中输入稀疏矩阵。例如，设文本文件 T.txt 中有三列内容 $\begin{bmatrix}1 & 3 & 5\\ 2 & 4 & 6\\ 2 & 5 & 8\\ 3 & 6 & 9\end{bmatrix}$，第一列是一些行下标，第二列是列下标，第三列是非零元素值，则利用 T.txt 创建稀疏矩阵：

```
load T.txt S=spconvert(T)
```

4）稀疏带状矩阵的创建

```
S=spdiags(B,d,m,n)
```

其中，m 和 n 分别是矩阵的行数和列数；d 是长度为 p 的整数向量，它指定矩阵 S 的对角线位置；B 是全元素矩阵，用来给定 S 对角线位置上的元素，行数为 min(m,n)，列数为 p。

5）其他稀疏矩阵创建函数

```
S=speye(m,n)
S=speye(size(A))      % 和 A 拥有同样尺寸的稀疏矩阵
S=buchy               % 一个内置的稀疏矩阵(邻接矩阵)
```

2. 稀疏矩阵的运算

稀疏存储矩阵只是矩阵的存储方式不同，其运算规则与普通矩阵相同，可以直接参与

运算。所以，MATLAB 中对满矩阵的运算和函数同样可用在稀疏矩阵中。结果是稀疏矩阵还是满矩阵，取决于运算符或函数。当参与运算的对象不全是稀疏存储矩阵时，所得结果一般是完全存储形式。

3. 其他

1）非零元素信息

```
nnz(S)        % 返回非零元素的个数
nonzeros(S) % 返回列向量，包含所有的非零元素
nzmax(S)      % 返回分配给稀疏矩阵中非零项的总的存储空间
```

2）查看稀疏矩阵的形状

```
spy(S)
```

3）find 函数与稀疏矩阵

```
[i,j,s]=find(S)
[i,j]=find(S)
```

返回 S 中所有非零元素的下标和数值，S 可以是稀疏矩阵或满矩阵。

2.4 多项式及其函数

若干个单项式的和组成的式子叫做多项式（在减法中，减一个数等于加上它的相反数）。多项式中每个单项式叫做多项式的项，这些单项式中的最高次数，就是这个多项式的次数。MATLAB 对于多项式的运算功能非常强大。

2.4.1 多项式的建立和操作

利用处理多项式的函数可以很方便求解多项式的根，并能很容易对多项式进行四则运算、积分和微分运算。

对于多项式 $P=a_0x^n+a_1x^{n-1}+a_2x^{n-2}+\cdots a_{n-1}x+a_0$ 约定可以用右边向量表示 $P=[a_0,a_1,a_2,\cdots,a_{n-1},a_n]$，这样多项式问题就转换为向量问题来解决。

（1）直接法创建多项式

【例 2-92】 直接法创建多项式。

```
>> P=[3 5 0 1 0 1 2]
P =
     3     5     0     1     0     1     2
>> y=poly2sym(P)
y =
    3*x^6+5*x^5+x^3+x+2
```

（2）指令 P=poly(AR)创建多项式

若已知多项式的全部根，则可以用 poly 函数建立起该多项式，也可以用 poly 函数求

矩阵的特征多项式。poly 函数是一个 MATLAB 程序，调用它的命令格式是

```
A=poly(x)
```

若 x 为具有 N 个元素的向量，则 poly(x)建立以 x 为其根的多项式，且将该多项式的系数赋值给向量 A。在此种情况下，poly 与 roots 互为逆函数；若 x 为 N×N 的矩阵 x，则 poly(x)返回一个向量赋值给 A，该向量的元素为矩阵 x 的特征多项式之系数：A(1),A(2),…,A(N),A(N+1)。

【例 2-93】 使用指令 P=poly(AR)创建多项式。

```
>> A=[3 1 4 1; 5 9 2 6; 5 3 5 8; 9 7 9 3]
A =
     3     1     4     1
     5     9     2     6
     5     3     5     8
     9     7     9     3
>> p=poly(A)
p =
    1.0000  -20.0000  -16.0000  480.0000   98.0000
```

（3）多项式的操作

- ❑ **roots(p)** 长度为 n 的向量，表示 n 阶多项式的根，即方程 p(x)=0 的根，可以为复数。
- ❑ **conv(p,q)** 表示多项式 p、q 的乘积，一般也指 p、q 的卷积。
- ❑ **poly(A)** 计算矩阵 A 的特征多项式向量。
- ❑ **poly(p)** 由长度为 n 的向量中的元素为根建立的多项式，结果是长度为 n+1 的向量。
- ❑ **polyval(p,x)** 若 x 为一数值，则计算多项式在 x 处的值；若 x 为向量，则计算多项式在 x 中每一元素处的值。

【例 2-94】 求特征方程的特征根。

```
>>  p=[3    0    2    3];
>> r=roots(p)           % rootp 为多项式的根
r =
   0.3911 + 1.0609i
   0.3911 - 1.0609i
  -0.7822
>> p=poly(r);
>> p
p =
    1.0000   -0.0000    0.6667    1.0000
```

2.4.2 多项式的计算

1. 多项式四则运算

多项式加减运算：MATLAB 没有提供专门进行多项式加减运算的函数，事实上，多项

式的加减就是其所对应的系数向量的加减运算。

对于次数相同的多项式，可以直接对其系数向量进行加减运算；如果两个多项式次数不同，则应该把低次多项式中系数不足的高次项用 0 补足，然后进行加减运算。

【例 2-95】 把多项式 a(x)与多项式 b(x)相加求解如下。

```
>> a=[1,2,3,4]
a =
     1     2     3     4
>> b=[4,5,6,7]
b =
     4     5     6     7
>> c=a+b
c =
     5     7     9    11
```

多项式乘法运算利用：

- **k=conv(p,q)** 事实上，多项式的相乘就是两个代表多项式的行向量的卷积。

【例 2-96】 计算多项式 $2x^3-x^2+3$ 和 $2x+1$ 的乘积。

```
>> p=[2,-1,0,3];
>> q=[2,1];
>> k=conv(p,q);
>> k
k =
     4     0    -1     6     3
```

多项式除法运算利用：

- **[k,r]=deconv(p,q)**

其中，k 返回的是多项式 p 除以 q 的商，r 是余式。

另外，其逆运算为：

- **p=conv(q,k)+r**

2. 多项式的导数

对于多项式求导应使用 polyder 函数

- **k=polyder(p)** %返回多项式 p 的一阶导数；
- **k=polyder(p,q)** %返回多项式 p 与 q 乘积的一阶导数；
- **[k,d]=polyder(p,q)** %返回 p/q 的导数，k 是分子，d 是分母。

【例 2-97】 已知 $p(x)=2x^3-x^2+3$， $q(x)=2x+1$， 求 $p',(p\cdot q)',(p/q)'$。

```
>> k1=polyder([2,-1,0,3])
k1 =
     6    -2     0
>> k2=polyder([2,-1,0,3],[2,1])

k2 =
    16     0    -2     6
```

```
>> [k2,d]=polyder([2,-1,0,3],[2,1])

k2 =
     8     4    -2    -6
d =
     4     4     1
```

3. 多项式求值

多项式求值函数 polyval 利用该函数可以求得多项式在某一点的值。

❑ **y=polyval(p,x)** 返回多项式 p 在 x 点的值，其中：x 可以是复数，也可以是矩阵。

【例 2-98】 已知 $p(x)=2x^3-x^2+3$，分别取 x=2 和一个 2×2 矩阵，求 $p(x)$在 x 处的值。

```
>> p=[2,-1,0,3];
>> x=2;polyval(p,x)
ans =
    15
>> x=[-1, 2;-2,1];polyval(p,x)
ans =
     0    15
   -17     4
```

4. 多项式求根

求解多项式的根，即 $p(x)$=0 的解。在 MATLAB 中，求解多项式的根有 roots 函数命令来完成。

❑ **x=roots(p)** 返回多项式的根，注意，按惯例，多项式是行向量，根是列向量。

【例 2-99】 已知 $p(x)=2x^3-x^2+3$，求 $p(x)$的根。

```
>> p=[2,-1,0,3];
>> x=roots(p)

x =
   0.7500 + 0.9682i
   0.7500 - 0.9682i
  -1.0000
```

若已知多项式的全部根，则可用 poly 函数给出该多项式。

$$p=ploy(x)\to p(x)=(x-x_1)(x-x_2)\cdots(x-x_n)$$

5. 有理多项式的部分分式展开

Residue 函数可以完成有理多项式的部分分式展开，它是一个对系统传递函数特别有用的函数，其调用格式如下：

格式一：

```
 [r,p,k]=residue(b,a)
```

功能：把 $b(s)/a(s)$展开成 $\dfrac{b(s)}{a(s)}=\dfrac{r_1}{s-p_1}+\dfrac{r_2}{s-p_2}+\ldots+\dfrac{r_n}{s-p_n}+k$，其中，$r$ 代表余数数组，

p 代表极点数组，k 代表常数项。

【例 2-100】 将有理多项式 $\frac{10s+20}{s^3+8s^2+19s+12}$ 展开成部分分式。

```
>> roum=[10,20]
roum =
    10    20
>> den=[1,8,19,12]
den =
     1     8    19    12
>> [r,p,k]=residue(roum,den)
r =
   -6.6667
    5.0000
    1.6667
p =
   -4.0000
   -3.0000
   -1.0000
k =
     []
```

即有理多项式可展开为 $\frac{-6.6667}{s+4}+\frac{5}{s+5}+\frac{1.667}{s+1}+0$ 。

格式二：

```
 [b,a]=residue(r,p,k)
```

功能：格式一的逆作用。

例如：[b,a]=residue(r,p,k)，其中，r,p,k 值同【例 2-100】。计算结果：

```
>> roum=[10,20]
roum =
    10    20
>> den=[1,8,19,12]
den =
     1     8    19    12
>> [r,p,k]=residue(roum,den)
>> [b,a]=residue(r,p,k)
     b=
       -0.0000  10.0000  20.0000
     a=
       1.0000  8.0000  19.0000  12.00000
```

6．特征多项式

格式：

```
poly(a)
```

（1）如果 a 是一个 n 阶矩阵，poly(a)是一个有 n+1 个元素的行向量，这 n+1 个元素是特征多项式的系数（降幂排列）。

（2）如果 a 是一个 n 维向量，则 poly(a)是多项式(x–a(1)) * (x–a(2)) *..(x–a(n))，即该多项式以向量 a 的元素为根。

7. 多项式曲线拟合

格式：

- **polyfit(x,y,n)**　polyfit(x,y,n)是找 n 次多项式 $p(x)$的系数，这些系数满足在最小二乘法意义下 p(x(i))≈y(i)。

2.5　本章小结

本章着重学习了 MATLAB 的几种重要数据类型及其操作函数。首先，简要学习了数组、矩阵、多项式的创建方法及操作函数。接着学习了数组、矩阵、多项式的运算方法。通过本章 MATLAB 的数组处理能力、M 函数指令、丰富的图形显示指令将使读者摆脱其他编程语言带来的编程烦恼。本章各节之间没有依从关系，却是全书学习的关键。

2.6　习题

（1）求解方程 $x^2-x-1=0$ 的根。

（2）输入矩阵 A=$\begin{bmatrix}1 & 2 & 3\\4 & 5 & 6\\7 & 8 & 9\end{bmatrix}$，使用全下标方式取出元素“3”，使用单下标方式取出元素“8”，取出后 2 行子矩阵块，使用逻辑矩阵方式取出$\begin{bmatrix}1 & 3\\7 & 9\end{bmatrix}$。

（3）输入 A 为 3×3 的魔方阵，B 为 3×3 的单位阵，由小矩阵组成 3×6 的大矩阵 C 和 6×3 的大矩阵 D，将 D 矩阵的最后 1 行构成小矩阵 E。

（4）输入字符串变量 a 为“hello”，将 a 的每个字符向后移 4 个，如“h”变为“l”，然后再逆序排放赋给变量 b。

（5）求矩阵$\begin{bmatrix}1 & 2\\3 & 4\end{bmatrix}$的转置矩阵、逆矩阵、矩阵的秩、矩阵的行列式值、矩阵的三次幂、矩阵的特征值和特征向量。

（6）求解方程组$\begin{cases}2x_1-3x_2+x_3+2x_4=8\\x_1+3x_2+x_4=6\\x_1-x_2+x_3+8x_4=7\\7x_1+x_2-2x_3+2x_4=5\end{cases}$

第 3 章　MATLAB 符号运算

符号计算是指利用数学定理和恒等式，通过推理和演绎，分析化简表达式，将复杂表达式变为形式简单的恒等表达式。利用符号运算可以避免计算过程中产生误差。MATLAB 中的符号计算功能是由 Maple 独立引擎提供的，利用这个内置的 Maple 符号计算引擎，可以进行各种针对符号对象或解析式的数学运算，如微积分运算，代数、微分方程求解，线性代数和矩阵运算，以及 Laplace 变换、Fourier 变换和 Z 变换。

3.1　符号运算入门

符号运算与数值计算相同，都是科学研究中的重要内容。运用符号运算，可以轻松解决许多公式和关系式的推导问题。

3.1.1　符号对象的创建

MATLAB 提供了两个建立符号对象的函数：sym 和 syms，这两个函数的用法不同。

1．sym 函数

sym 函数用来创建单个符号量，调用格式为

```
符号量名=sym('符号字符串')
```

该函数可以建立一个符号量，符号字符串可以是常量、变量、函数或表达式。

【例 3-1】 利用 sym 函数创建符号变量，完成对方程组求解。

```
>> a=sym('a');
>> b=sym('b');
>> x=sym('x');
>> y=sym('y');
>> [x,y]=solve('a*x-b*y=1','a*x+b*y=3','x','y')
 x =
 2/a
 y =
 1/b
 >>
```

【例 3-2】 创建符号变量，求复数表达式 z=x+i*y 的共轭复数。

```
>> x=sym('x','real');
>> y=x+i*y;
>> x=sym('x','real');
>> y=sym('y','real');
>> z=x+i*y;
```

```
>> conj(z)
   ans =
x - y*i
```

2．syms 函数

syms 函数可以在一条语句中定义多个符号变量，调用格式为

```
syms  符号变量名 1 符号变量名 2 … 符号变量名 n
```

用这种格式定义符号变量时不要在变量名上加字符串分界符“‘”，变量间用空格而不要用逗号分隔。在数学表达式中，一般习惯于使用排在字母表中前面的字母作为变量的系数，而用排在后面的字母表示变量。

例如，f=ax^2+bx+c，表达式中的 a、b、c 通常被认为是常数，用作变量的系数；而将 x 看作自变量。若在 MATLAB 中表示上述表达式，首先用 syms 函数定义 a、b、x 为符号对象。在进行导数运算时，由于没有指定符号变量，则系统采用数学习惯来确定表达式中的自变量，默认 a、b、c 为符号常数，x 为符号变量。即对函数 f 求导为 df/dx。

3.1.2 符号表达式的创建

含有符号对象的表达式被称为符号表达式，一个符号表达式应该由符号变量、函数、算术运算符组成，符号表达式的书写格式与数据表达式相同。

表 3-1　符号表达式和 MATLAB 表达式的对照

符号表达式	MATLAB 表达式
$y=\frac{1}{\sqrt{2x}}$	y='1/sqrt(2*x)'
$\cos(x^2)-\sin(2x)$	'cos(x^2)-sin(2*x)'
$\frac{e^{x^3}}{\sqrt{1-x}}$	'exp(x^3)/sqrt(1-x)'
$\frac{1}{2x^n}$	'1/(2*x^n)'

符号表达式有如下三种建立的办法。

（1）利用单引号来生成符号表达式。

（2）用 sym 函数建立符号表达式。

（3）使用已经定义的符号变量组成符号表达式。

将表达式中的自变量定义为符号变量后，赋值给符号函数名，即可生成符号函数。例如有一数学表达式，其用符号表达式生成符号函数 fxy 的过程为

```
fxy=(a*x^2+b*y^2)/c^2          %生成符号函数
syms a b c x y                 %定义符号运算量
```

生成符号函数 fxy 后，即可用于微积分等符号计算。

【例 3-3】 符号函数 fxy=(a*x2+b*y2)/c2 ，分别求该函数对 x、y 的导数和对 x 的积分。

```
syms a b c x y                 %定义符号变量
```

```
fxy=(a*x^2+b*y^2)/c^2;       %生成符号函数
diff(fxy,x)                  %符号函数 fxy 对 x 求导数
ans =2*a*x/c^2
diff(fxy, y)                 %符号函数 fxy 对 y 求导数
ans =2*b*y/c^2
int(fxy, x)                  %符号函数 fxy 对 x 求积分
ans =1/c^2* (1/3*a*x^3+b*y^2*x)
```

3.1.3 符号矩阵的相关操作

1. 使用 sym 函数创建

sym 函数可以创建符号矩阵，用法如下。

```
A=sym('[ ]')
```

【例 3-4】 利用 sym 函数直接创建符号矩阵。

```
A = sym('[a , 2*b ; 3*a , 0]')
A =
    [ a, 2*b]
    [3*a, 0]
```



符号矩阵的每一行的两端都有方括号，这是与 MATLAB 数值矩阵的一个重要区别。

2. 基于字符串创建

（1）用字符串直接创建矩阵。

（2）模仿 MATLAB 数值矩阵的创建方法。

需保证同一列中各元素字符串有相同的长度。

【例 3-5】 模仿数值矩阵的方式创建符号矩阵。

```
A =['[ a,2*b]'; '[3*a, 0]']
A =
    [ a, 2*b]
    [3*a, 0]
```

3. 符号矩阵的修改

修改符号矩阵有两种方式，利用光标键移到指定位置直接修改或利用指令修改，本段主要介绍用指令修改。

指令修改运用了 subs 函数，使用形式如

```
A1=subs(A, 'new', 'old')
```

【例 3-6】 利用 subs 函数修改符号矩阵。

```
A =[ a, 2*b]
  [3*a, 0]
A = sym('[a , 2*b ; 3*a , 0]')
```

```
A1=sym(A,2,2,'4*b') %%等效于 A(2,2)='4*b';
A1 =[ a, 2*b]
    [3*a, 4*b]
A1=subs(A,'0','4*b')
A2=subs(A1, 'c', 'b')
A2 =[ a, 2*c]
    [3*a, 4*c]
```

3.1.4 符号运算中的运算符

MATLAB 中为符号运算提供了多种多样的运算符，如表 3-2 所示

表 3-2 符号运算中的运算符

符 号	符号用途说明
+	加
-	减
.*	点乘
*	矩阵相乘
^	矩阵求幂
.^	点幂
\	左除
/	右除
.\	点左除
./	点右除
kron	张量积
,	分隔符
;	（a）写在表达式后面时运算后不显示计算结果 （b）在创建矩阵的语句中指示一行元素的结束，如 m=[x y z;i j k]
:	创建向量的表达式分隔符，如 x=a:b:c a(:,j)表示 j 列的所有行元素；a(i,:)表示 i 行的所有列元素
[ ]	创建数组、向量、矩阵或字符串（字母型）
{ }	创建单元矩阵或结构
%	注释符，特别当编写自定义函数文件时，紧跟 function 后的注释语句， 在你使用 help 函数名时会显示出来
'	（a）定义字符串用 （b）向量或矩阵的共轭转置符
.'	一般转置符
...	表达式换行标记，表示表达式继续到下一行
=	赋值符号
==	等于关系运算符
<,>	小于，大于关系运算符
&	逻辑与
|	逻辑或
~	逻辑非
xor	逻辑异或

3.1.5 符号表达式中自变量的确定

MATLAB 中的符号可以表示符号变量和符号常量，findsym 可以帮助用户查找一个符号表达式中的符号变量，其调用方法如下。

- **findsym(expr)** 确定表达式 expr 中的所有符号为自变量。
- **findsym(expr,n)** 确定表达式 expr 中靠 x 最近的 n 个自变量。

【例 3-7】 利用 findsym 确定表达式中的自变量。

```
syms a x y z t
findsym(sin(pi*t))
findsym(x+i*y-j*z,1)
findsym(x+i*y-j*z,2)
findsym(x+i*y-j*z,3)
syms x a y z b;                %定义 5 个符号变量
s1=3*x+y;s2=a*y+b              %定义两个符号表达式
findsym(s1)
findsym(s2,2)
syms x y;
s=2*x+3*y;
findsym(s)
ans =
      x, y
syms a b x y;                  %定义符号变量
c=sym('3');                    %定义符号常量 c
findsym(a*x+b*y+c)
ans =                          %c 不在结果中出现
      a, b, x, y
```



MATLAB 按离字母 x 最近原则确定默认变量。

3.2 符号表达式运算

符号表达式可以进行多种运算，如基本的四则运算，也可进行表达式求值、数值转换及变量替换。

3.2.1 提取分子和分母

如果表达式是一个有理分式（两个多项式之比），或可以展开为有理分式（包括哪些分母为 1 的分式），可以利用 numden 将分子或分母提取出来。

【例 3-8】 利用 numden 提取分子分母。

```
 >> m= ' x^2 '
m=       x^2
```

```
 >> [n, d]=numden(m)
n=       x^2
d=     1
 >> f= ' a*x^2/(b-x) '
f=       a*x^2/(b-x)
 >> [n, d]=numden(f)
n=       a*x^2
d=       b-x
 %前二个表达式得到期望结果。
 >> g= ' 3/2*x^2+2/3*x-3/5 '
g=       3/2*x^2+2/3*x-3/5
 >> [n, d]=numden(g)
n=       45*x^2+20*x-18
d=       30
 >> h= ' (x^2+3)/(2*x-1)+3*x/(x-1) '
h=        (x^2+3)/(2*x-1)+3*x/(x-1)
 >> [n, d]=numden(h)
n=       x^3+5*x^2-3
d=        (2*x-1) * (x-1)
```

%在提取各部分之前，这两个表达式 g 和 h 被有理化，并变换成具有分子和分母的一个简单表达式。

```
>> k=sym( ' [3/2, (2*x+1)/3; 4/x^2, 3*x+4] ' ) %  try a symbolic array
k=
  [   3/2, (2*x+1)/3]
  [4/x^2,      3*x+4]
>> [n, d]=numden(k)
n=
  [3,  2*x+1]
  [4,  3*x+4]
d=
  [  2, 3]
  [x^2, 1]
```

这个表达式 k 是符号数组，numden 返回两个新数组 n 和 d，其中，n 是分子数组，d 是分母数组。如果采用 s=numden(f)形式，numden 仅把分子返回到变量 s 中。

numden 也可以化简分数表达式如下。

```
>> syms x y;
>> f=x/y+y/x;
>> [n,d]=numden(f)
 n =
     x^2 + y^2
 d =
     x*y
```

3.2.2 数值转换

1. 数据类型转换函数

利用数据类型转换函数可以将常数转换为数值，常用的数据类型转换函数如表 3-3

所示。

表 3-3　数据类型转换函数

函　数　名	作　　用
logical	数值转化为逻辑值
char	转换为字符串数组
int8	转换为 8 字节整型数
uint8	转换为 8 字节数
int16	转换为 16 字节整型数
uint16	转换为 16 字节数
int32	转换为 32 字节整型数
uint32	转换为 32 字节数
int64	转换为 64 字节整型数
uint64	转换为 64 字节数
single	转换为单精度浮点数
double	转换为 64 字节浮点数
cell	转换为细胞数组
struct	转换为结构体类型

【例 3-9】 利用转换函数转换符号常量。

```
>> a=3.8495;
>> f=sym('6*a+2^(2*a)');
>> m=eval(f)
m =
   230.8895
>> int8(m)
ans =
   127
>> logical(m)
ans =
     1
```

2．sym2poly

sym2ploy 可以将符号表达式转换为数值多项式的系数向量，且系数从高到低依次排列。

【例 3-10】 使用 sym2poly 函数显示数值多项式的系数向量。

```
f=sym(7+3*x+5*x^2);
sym2poly(f)
ans=
     5   3   7
```

3．poly2sym

poly2sym 与 sym2poly 相反，可以将数值多项式的系数向量转换为符号表达式。

```
a=[5 3 7];
```

```
poly2sym(a)
ans=
     5*x^2+3*x+7
```

4. eval

MATLAB 中的 eval 函数可以计算符号表达式的具体值。

【例 3-11】 计算符号表达式 k*2+2^m 的值。

```
>> f=sym('k*2+2^m');
>> k=sym('5');
>> m=sym('7');
>> r=eval(f)
 r =
    138
```

3.2.3 变量替换

MATLAB 中提供了 subs 函数用于实现变量的替换，在处理复杂函数方程式时会使计算更简便。

- **Subs(S,old,new)** 用 new 替换 S 中的 old 变量，old 必须是 S 中的符号变量。
- **Subs(S,new)** 用 new 替换 S 中的自变量。

【例 3-12】 subs 函数用于实现变量的替换。

```
>> syms a m n w;
>> f=2+3^a;
>> subs(f,'a','m^2+5*n+w')
ans =
      3^(m^2 + 5*n + w) + 2
```

3.2.4 化简与格式化

MATLAB 提供了多种函数来实现对符号运算表达式进行化简，如 factor（因式分解）、collect（合并同类项）、horner（将多项式分解为嵌套形式）、expand（展开表达式为多项式、指数函数、对数函数、三角函数）、simplify（化简一个表达式）、simple（将表达式化到最简形式）。

1. 因式分解 factor

- **factor(x)** 若 x 可分解时，返回分解后的表达式，否则，返回原 x。

【例 3-13】 利用 factor 分解表达式 x^2+4*x+5。

```
>> f=sym('x^2+4*x+5');
>> factor(f)
ans =
      x^2 + 4*x + 5
      syms a b x y;
      A=a^3-b^3;
      factor(A)
```

```
ans =
    (a-b) * (a^2+a*b+b^2)
>>syms x; f=x^6+1;
>>factor(f)
  % factor 也可用于正整数的分解
 >>s=factor(100)
 >>factor(sym('12345678901234567890'))
```

大整数的分解要转化成符号常量，否则，表达精度不够会出错。

2. 合并同类项 collect

- ❑ **collect(S)**　将 S 中相同次幂的项合并，S 可以是表达式也可以是符号矩阵。
- ❑ **collect(S,v)**　将 S 中 v 的相同幂次的项进行合并。

【例 3-14】 使用 collect 函数实现合并同类项。

（1）使用 collect 函数的第一种形式合并同类项。

```
>> f=sym('(x^2+2*x) * (x+2)');
>> collect(f)
 ans =
       x^3 + 4*x^2 + 4*x
```

（2）使用 collect 函数的第二种形式合并同类项。

```
>> syms x y;
>> f=(y^3+2*x) * (x+5);
>> collect (f,x)
 ans =
       2*x^2 + (y^3 + 10) *x + 5*y^3
>> syms x y;
>> f= x^2*y + y*x - x^2 + 2*x ;
>> collect(f)
>> collect(f,y)
 ans =
       (y - 1) *x^2 + (y + 2) *x
 ans =
       (x^2 + x) *y + 2*x - x^2
```

3. 多项式分解 horner

用法：horner(S)，S 是符号多项式矩阵，horner 函数可以将每个多项式转换成嵌套形式。

【例 3-15】 分解多项式 f=5*x^4+3*x^2-x。

```
>> syms x
>> f=5*x^4+3*x^2-x;
>> horner(f)
ans =

       x* (x* (5*x^2 + 3) - 1)
>> syms x;
>> f=x^4+2*x^3+4*x^2+x+1;
>> g=horner(f)
```

```
    g =
       x* (x* (x* (x + 2) + 4) + 1) + 1
```

4. 展开表达式 expand

用法：expand(S)，若 S 是多项式，则展开为相应的形式；若 S 是三角函数、指数函数和对数函数，则根据要求展开成相应形式。

【例 3-16】 用 expand 函数展开多项式。

```
>> syms x,y;
>> f=(5*x+4*y+3)^2;
>> expand(f)
    ans =
        25*x^2 + 40*x*y + 30*x + 16*y^2 + 24*y + 9
    s=(-7*x^2-8*y^2) * (-x^2+3*y^2)
    expand(s)                          %对 s 展开
    ans =
        7*x^4-13*x^2*y^2-24*y^4
%多项式展开
>> syms x;
>> f=(x+1)^6;
>> expand(f)
    ans =
       x^6 + 6*x^5 + 15*x^4 + 20*x^3 + 15*x^2 + 6*x + 1
%三角函数展开
>> syms x y; f=sin(x+y);
   expand(f)
   ans =
       cos(x)*sin(y) + cos(y)*sin(x)
```

5. 化简表达式 simplify

用法：simplify(S)，表达式 S 可以是多项式也可以是符号表达式矩阵。

【例 3-17】 用 simplify 函数化简多项式。

（1）化简 sin(x)^2+cos(x)^2+2*Sin(x) *cos(x)。

```
>> f=sym('sin(x)^2+cos(x)^2+2*Sin(x)*cos(x)');
>> simplify(f)
    ans =
         2*Sin(x)*cos(x) + 1
```

（2）化简 log(2*x/y)和(-a^2+1)/(1-a)。

```
>>syms x y a
>>s=log(2*x/y);
>>simplify(s)
   ans =
         log(2)+log(x/y)
>>s=(-a^2+1)/(1-a)
>>simplify(s)
   ans =
         a+1
>>syms x;
```

```
>>f=sin(x)^2 + cos(x)^2 ;
>>simplify(f)
 ans =
       1
>> syms c alpha beta;
>> f=exp(c*log(sqrt(alpha+beta)));
>> simplify(f)
 ans =
     (alpha + beta)^(c/2)
```

6. 最简转化 simple

有如下两种用法：

- **[r,how]=simple(S)** 返回值 r 是最简形式的符号表达式，how 是描述简化过程的字符串。
- **r=simple(S)** simple 函数将显示表达式 S 所有的简化形式，并返回其中最短的一个。

【例 3-18】 用 simple 函数化简多项式。

（1）化简 $f(x)=\sqrt[3]{\frac{1}{x^3}+\frac{6}{x^2}+\frac{12}{x}+8}$ 。

```
>> syms x;
>> f=(1/x^3+6/x^2+12/x+8)^(1/3);
>> y1=simplify(f)
   y1 =
       ((2*x + 1)^3/x^3)^(1/3)
 %多次使用 simple 可以达到最简表达。
>> g1=simple(f)
   g1 =
        ((2*x + 1)^3/x^3)^(1/3)
 >> g2=simple(g1)
    g2 =
        ((2*x + 1)^3/x^3)^(1/3)
```

（2）化简 y=(2+x)/x。

```
>> s=sym('y=(2+x)/x');
>> [r,how]=simple(s)
    r =
      x + 2 = x*y
    how =
      simplify(100)
```

（3）化简 2*sin(x) *cos(x)。

```
>> y=sym('2*sin(x) *cos(x)');
>> simple(y)
    simplify:
    sin(2*x)
    radsimp:
    2*cos(x) *sin(x)
    simplify(100):
```

```
sin(2*x)
combine(sincos):
sin(2*x)
combine(sinhcosh):
2*cos(x)*sin(x)
combine(ln):
2*cos(x)*sin(x)
factor:
2*cos(x)*sin(x)
expand:
2*cos(x)*sin(x)
combine:
2*cos(x)*sin(x)
rewrite(exp):
2*((1/exp(x*i))/2 + exp(x*i)/2)*(((1/exp(x*i))*i)/2 - (exp(x*i)*i)/2)
rewrite(sincos):
2*cos(x)*sin(x)
rewrite(sinhcosh):
2*cosh(-x*i)*sinh(-x*i)*i
rewrite(tan):
-(4*tan(x/2)* (tan(x/2)^2 - 1))/(tan(x/2)^2 + 1)^2
mwcos2sin:
-2*sin(x)* (2*sin(x/2)^2 - 1)
collect(x):
2*cos(x)*sin(x)
ans =
     sin(2*x)
```

3.2.5 数值表达式和符号表达式的互相转换

利用函数 sym 可以将数值表达式变换为其符号表达式。

```
>>sym(1.5)
  ans =
       3/2
>> sym(4.45)
  ans =
      89/20
```

函数 numeric 或 eval 可以将符号表达式变换成数值表达式：

```
phi='(1+sqrt(5))/2'
phi =
     (1+sqrt(5))/2
     numeric(phi)
ans =
     1.6180
>> p='(1+2^3)/2'
  p =
     (1+2^3)/2
```

```
>> eval(p)
   ans =
         4.5000
```

3.2.6 反函数

在 MATLAB 中，可以使用 finverse 计算反函数。

- **g=finverse(f)** 返回符号函数 f 的反函数 g。其中，f 是一个符号函数表达式，其变量为 x。求得的反函数 g 是一个满足 g(f(x))=x 的符号函数。

```
>> syms x;
>> f=sym(2/sin(x));
>> finverse(f)
   ans =
         asin(2/x)
```

- **g=finverse(f,v)** 返回自变量 v 的符号函数 f 的反函数。求得的反函数 g 是一个满足 g(f(v))=v 的符号函数。当 f 包含不止一个符号变量时，往往调用这个格式。

当 finverse 求得的解不唯一时，matlab 会给出警告。

```
>> syms x;
>> f=sym(x^2+1);
>> finverse(f)
   ans =
         (-1+x)^(1/2)
```

3.2.7 表达式替换函数

（1）subs(s) 用赋值语句中给定值替换表达式中所有同名变量。

（2）subs (s,old,new) 用符号或数值变量 new 替换原来 s 中的符号变量 old。

常用格式如下。

- **subs(f)** 求符号表达式 f 的值。
- **subs(f,a)** 用 a 替换 f 中的默认变量 x，并求值。
- **subs(f,x,a)** 用 a 替换 f 中的指定变量 x，并求值。

【例 3-19】 表达式替换函数演示。

（1）将表达式 x^2+y^2 中 x 取值为 2。

```
syms x y;
f=x^2+y^2;
subs(f,x,2)
```

（2）同时对两个或多个变量取值求解。

```
syms x y ;
f=x^2+y^2;
subs(f,[x,y],[1,2])    %同时替换两个变量并求值
```

```
subs(f,[x,y],[a+b,a-b])  %方括号换成大花括号也可以
```

3.3 符号运算精度

MATLAB 提供了三种计算精度：浮点运算的数值算法，精确运算的符号算法和可控精度的算法。

1. 浮点运算的数值算法

浮点运算的数值算法是运算速度最快的运算方法，由于在计算机中以二进制进行存储，计算时取近似值，不可避免地会产生误差。

【例 3-20】 浮点运算的数值算法样例。

```
>> sym a;
>> a=2/3+4/7
  a =
      1.2381
```

2. 精确运算的符号算法

精确运算速度较慢，但精确。

【例 3-21】 精确运算的符号算法样例。

```
>> a=sym(2/3+4/7)
  a =
      26/21
```

3. 可控精度的算法

可控精度的算法通过规定有效数字位数控制精度，位数不同精度也不同。

- ❑ **digits(n)** 规定参加运算有效数字的位数，MATLAB 默认值为 32。
- ❑ **vpa(s)** 在 digits(n)控制下计算指定精度的 s，如果 n 未指定则默认 32。

【例 3-22】 可控精度的算法样例。

```
>> syms a b c
>> a=1/3+5/7;
>> b=pi;
>> c=3.7878882;
>> d=sym(4/9);
>> f1=vpa(a+b)
   f1 =
        4.1892
>> f2=vpa(a+c)
   f2 =
        4.8355
 >> f3=vpa(a+d)
    f3 =
        1.4921
>> digits(20)
>> f4=vpa(a+b)
```

```
   f4 =
        4.1892117012088405659
>> f5=vpa(a+c)
    f5 =
        4.8355072476190477104
>> f6=vpa(a+d)
    f6 =
        1.4920634920634920635
```

3.4 符号矩阵的计算

在进行符号矩阵的计算时，很多方面在形式上和数值计算相同，不必再去重新学一套关于符号运算的新规则。这里介绍的符号矩阵运算在形式上与数值计算中的运算相似，容易掌握。

3.4.1 基本代数运算

在 MATLAB 中，符号对象的代数运算和双精度运算从形式上看是相同的，由于 MATLAB 中采用了符号的重载，用于双精度运算的运算符同样可以用于符号对象。

【例 3-23】 符号矩阵的加减运算输入指令如下。

```
>> syms a b c d;                  %定义基本的符号变量
>> A=sym('[a b;c d]');            %定义符号矩阵
>> B=sym('[a 2*b;c+b d-2]');      %定义符号矩阵
>> A+B;                           %计算符号矩阵的加法
>> A-B;                           %计算符号矩阵的减法
>> A*B;                           %计算符号矩阵的乘法
>> A/B;                           %计算符号矩阵的除法
>> syms a b c d;                  %定义基本的符号变量
```

输出结果如下：

```
ans =
      [   2*a,   3*b]
      [ 2*c+b, 2*d-2]
ans =
      [  0, -b]
      [ -b,  2]
ans =
      [a^2+b* (c+b), 2*a*b+b* (d-2)]
      [c*a+d* (c+b), 2*c*b+d* (d-2)]
ans =
      [ (a*d-2*a-c*b-b^2)/(-2*c*b-2*b^2+a*d-2*a),
      -a*b/(-2*c*b-2*b^2+a*d-2*a)]
      [ -(2*c+d*b)/(-2*c*b-2*b^2+a*d-2*a),
      (a*d-2*c*b)/(-2*c*b-2*b^2+a*d-2*a)]
```

【例 3-24】 计算符号矩阵的二次方、三次方运行输入指令如下。

```
>> A=sym('[7 4 2;1 5 6;3 0 8]');     %定义符号矩阵
>> A^2                               %计算符号矩阵的二次方
>>A^3                                %计算符号矩阵的三次方
```

输出结果如下。

```
A =
     [ 7, 4, 2]
     [ 1, 5, 6]
     [ 3, 0, 8]
ans =
     [ 59, 48, 54]
     [ 30, 29, 80]
     [ 45, 12, 70]
 ans =
     [ 623, 476, 838]
     [ 479, 265, 874]
     [ 537, 240, 722]
```

【例 3-25】 计算符号矩阵的 4 次幂，输入指令如下所示。

```
>> A=sym('[1 2 3;4 5 6;7 8 9]');           %定义符号矩阵
>>A^4
```

输出结果如下：

```
A =
   [ 1, 2, 3]
   [ 4, 5, 6]
   [ 7, 8, 9]
ans =
   [  7560,  9288, 11016]
   [ 17118, 21033, 24948]
   [ 26676, 32778, 38880]
```

【例 3-26】 计算符号矩阵的指数，输入指令如下所示。

```
>> A=sym('[1 2 3;4 5 6;7 8 9]');           %定义符号矩阵
>>B=exp(A)
```

输出结果如下：

```
A =
    [ 1, 2, 3]
     [ 4, 5, 6]
     [ 7, 8, 9]
B =
     [ exp(1), exp(2), exp(3)]
     [ exp(4), exp(5), exp(6)]
     [ exp(7), exp(8), exp(9)]
```

3.4.2 线性代数运算

符号对象的线性代数运算和双精度的线性代数运算相同，有以下几种函数指令，如表 3-4 所示。

表 3-4 符号矩阵线性运算函数

函 数 名 称	功 能 介 绍
inv	矩阵求逆
det	计算行列式的值
diag	对角矩阵
triu	抽取矩阵的上三角部分
tril	抽取矩阵的下三角部分
rank	计算矩阵的秩
rref	返回矩阵的所见行阶梯矩阵
null	零空间的正交基
colspace	返回矩阵列空间的基
transpose	返回矩阵的转置
eig	特征值分解
jordan	约当标准型变换
svd	奇异值分解

函数的具体用法如下。

1. inv 函数

MATLAB 提供了 inv 函数指令用于计算符号矩阵的逆，其具体用法如下。

- **inv(A)** 计算符号矩阵的逆。

【例 3-27】 生成数值希尔伯特矩阵，计算其逆矩阵输入指令如下。

```
>> A=hilb(4);          %定义符号矩阵
>>A=sym(A);
>>inv(A)
```

输出结果如下。

```
A =
   [  1, 1/2, 1/3, 1/4]
   [ 1/2, 1/3, 1/4, 1/5]
   [ 1/3, 1/4, 1/5, 1/6]
   [ 1/4, 1/5, 1/6, 1/7]
ans =
      [   16,  -120,   240,  -140]
      [ -120,  1200, -2700,  1680]
      [  240, -2700,  6480, -4200]
      [ -140,  1680, -4200,  2800]
```

2. det 函数

MATLAB 提供了 det 函数指令用于计算符号矩阵的行列式，其具体用法如下。

❑ **det(A)** 计算矩阵 A 的行列式。

【例 3-28】 计算【例 3-5】中符号矩阵的行列式，输入指令如下。

```
>> A=hilb(4);          %定义符号矩阵
>>A=sym(A);
>>det(A)
```

输出结果如下。

```
A =
   [  1, 1/2, 1/3, 1/4]
   [ 1/2, 1/3, 1/4, 1/5]
   [ 1/3, 1/4, 1/5, 1/6]
   [ 1/4, 1/5, 1/6, 1/7]
ans =
    1/6048000
```

3．diag 函数

diag 函数指令用于实现对符号矩阵对角线元素的操作，其具体用法如下。

❑ **diag(v,k)** 若 v 是由 n 个元素构成的矢量，则结果是 n+abs(k)阶方阵，当 k=0 时，将矢量 v 至于主对角线上；当 k<0 时，将矢量 v 置于主对角线之下；当 k>0 时，将矢量 v 置于主对角线之下。

❑ **diag(v)** 与 k=0 相同，将矢量 v 置于主对角线上。

❑ **diag(A,k)** A 是矩阵，结果是由矩阵 A 的第 k 条对角线上的元素组成的列矢量。

❑ **diag(A)** A 是矩阵，是 diag(A,k)用法中 k=0 的情况，结果是由矩阵 A 的主对角线元素组成的列矢量。

【例 3-29】 如果 a 是由四个元素构成的矢量，利用 diag 函数指令求解符号矩阵的对角线，输入指令如下。

```
>>  syms a b c;          %定义符号矩阵
>>  A=[a b+1 c*3 4];
>>  diag(A,1)
```

输出结果如下。

```
ans =
      [ 0, a,     0,   0, 0]
      [ 0, 0, b + 1,   0, 0]
      [ 0, 0,     0, 3*c, 0]
      [ 0, 0,     0,   0, 4]
      [ 0, 0,     0,   0, 0]
```

【例 3-30】 利用 diag 函数指令将矢量 a 置于主对角线上，输入指令如下。

```
>> syms a b c;          %定义符号矩阵
>>A=[a b+1 c*3 4]
>>diag(A)
```

输出结果如下：

```
A =
    [ a, b + 1, 3*c, 4]
ans =
      [ a,     0,   0, 0]
      [ 0, b + 1,   0, 0]
      [ 0,     0, 3*c, 0]
      [ 0,     0,   0, 4]
```

【例 3-31】 假设 A 为四阶希尔伯特矩阵，利用 diag 函数指令求矩阵的对角线，输入指令如下。

```
>>A=hilb(4)
>>diag(A)
```

输出结果如下。

```
A =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
ans =
    1.0000
    0.3333
    0.2000
    0.1429
```

【例 3-32】 A 是随机矩阵，分别找出由矩阵 A 的第 1、2、3、4 条对角线上的元素组成的列矢量，输入指令如下。

```
>> s=sym(4);          %定义符号矩阵
>> class(s);
>> A=rand(s);
>> diag(A,3);
>> diag(A,2);
>> diag(A,1);
>> diag(A,0)
```

输出结果如下。

```
A =
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
ans =
    0.9572
ans =
    0.9575
    0.4854
ans =
```

```
        0.6324
        0.9649
        0.8003
    ans =
        0.8147
        0.0975
        0.1576
        0.1419
```

【例 3-33】 假设 A 是一个三阶魔方矩阵，利用 diag 函数指令找出矩阵 A 主对角线上元素的列矢量，输入指令如下。

```
>> s=sym(4);          %定义符号矩阵
>> class(s);
>> A=magic(s);
>> diag(A)
```

输出结果如下。

```
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
ans =
    16
    11
     6
     1
```

4. triu 函数

MATLAB 提供 triu 函数对符号矩阵的上三角部分进行操作，其具体用法如下。

- ❑ **triu(A)** 抽取矩阵 A 主对角线上的三角部分重新组成一个新矩阵，其他部分用 0 来填充。
- ❑ **triu(A,k)** 抽取矩阵 A 的第 k 条对角线上的部分重新组成一个新矩阵，其他部分用 0 来填充。当 k>0 时，抽取的元素是在主对角线上且在 k 条对角线上的元素，其他部分用 0 来填充。当 k<0 时，抽取的元素是在主对角线下且在 k 条对角线上的元素，其他部分用 0 来填充。当 k=0 时，即 triu(A,0),与 triu(A)相同，抽取主对角线上的部分。

【例 3-34】 假设 A 是一个五阶魔方矩阵，利用 triu 函数指令生成一个由矩阵 A 主对角线上的元素组成的矩阵，输入指令如下。

```
>> s=sym(5);          %定义符号矩阵
>> class(s);
>> A=magic(s);
>> triu(A)
```

输出结果如下。

```
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
ans =
    17    24     1     8    15
     0     5     7    14    16
     0     0    13    20    22
     0     0     0    21     3
     0     0     0     0     9
```

【例 3-35】 假设 A 是一个五阶魔方矩阵，利用 triu 函数指令生成由矩阵 A 主对角线上的元素组成的矩阵，以及由矩阵 A 主对角线下的元素组成的矩阵和由矩阵 A 主对角线的元素组成的矩阵，输入指令如下。

```
>> s=sym(5);          %定义符号矩阵
>> class(s);
>> A=magic(s);
>> triu(A,2);
>> triu(A,-2);
>> triu(A,0);
```

输出结果如下：

```
A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
ans =
     0     0     1     8    15
     0     0     0    14    16
     0     0     0     0    22
     0     0     0     0     0
     0     0     0     0     0
ans =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
     0    12    19    21     3
     0     0    25     2     9
ans =
    17    24     1     8    15
     0     5     7    14    16
     0     0    13    20    22
     0     0     0    21     3
     0     0     0     0     9
```

【例 3-36】 假设 A 是一个三阶随机矩阵，利用 triu 函数指令生成由矩阵 A 主对角线上的元素组成的矩阵，以及由矩阵 A 主对角线下的元素组成的矩阵和由矩阵 A 主对角线的元素组成的矩阵，输入指令如下。

```
>> s=sym(3);          %定义符号矩阵
>> class(s);
>> A=rand(s);
>> triu(A,1);
>> triu(A,-1);
>> triu(A,0);
```

输出结果如下。

```
A =
    0.4218    0.9595    0.8491
    0.9157    0.6557    0.9340
    0.7922    0.0357    0.6787
ans =
         0    0.9595    0.8491
         0         0    0.9340
         0         0         0
ans =
    0.4218    0.9595    0.8491
    0.9157    0.6557    0.9340
         0    0.0357    0.6787
ans =
    0.4218    0.9595    0.8491
         0    0.6557    0.9340
         0         0    0.6787
```

【例 3-37】 利用 triu 函数生成由符号矩阵 A 主对角线上的元素组成的矩阵，以及由符号矩阵 A 主对角线下的元素组成的矩阵和由符号矩阵 A 主对角线的元素组成的矩阵，对比它们的不同，输入指令如下。

```
>> syms a b c;          %定义符号矩阵
>> A=[a^2 b+c 6 exp(c);a+b b a 5;4 b c 1;a^b c a 8]
>> triu(A);
>> triu(A,1);
>> triu(A,-1)
```

输出结果如下。

```
A =
   [   a^2, b + c,  6,exp(c)]
   [  a +b,     b, a,      5]
   [     4,     b, c,      1]
   [   a^b,     c, a,      8]
ans =
   [   a^2, b + c,  6,exp(c)]
   [     0,     b, a,      5]
   [     0,     0, c,      1]
```

```
        [ 0,    0, 0,     8]
    ans =
        [ 0, b +c, 6,exp(c)]
        [ 0,    0, a,     5]
        [ 0,    0, 0,     1]
        [ 0,    0, 0,     0]
    ans =
        [   a^2, b + c, 6,exp(c)]
        [  a +b,     b, a,     5]
        [     0,     b, c,     1]
        [     0,     0, a,     8]
```

5. tril 函数

MATLAB 提供了 tril 函数生成一个新矩阵，该新矩阵式抽取原矩阵的下三角部分，其他部分用 0 来填充，其具体用法如下。

- **tril(A)** 抽取矩阵 A 的主对角线下的三角部分重新组成一个新矩阵，其他部分用 0 来填充。
- **tril(A,k)** 抽取矩阵 A 的第 k 条对角线下的部分重新组成一个新矩阵，其他部分用 0 来填充。当 k>0 时，抽取的元素是在主对角线上且 k 条对角线下的元素，其他部分用 0 来填充。当 k<0 时，抽取的元素是在主对角线下且 k 条对角线下的元素，其他部分用 0 来填充。当 k=0 时，即 tril(A,0),与 tril 相同，抽取主对角线下的部分。

【例 3-38】 利用 tril 函数生成由矩阵 A 主对角线下的元素所组成的矩阵，输入指令如下。

```
>> A=magic(4)
>> B=sym(A)
>> tril(B)
```

输出结果如下。

```
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
B =
     [ 16,  2,  3, 13]
     [  5, 11, 10,  8]
     [  9,  7,  6, 12]
     [  4, 14, 15,  1]
ans =
     [ 16,  0,  0, 0]
     [  5, 11,  0, 0]
     [  9,  7,  6, 0]
     [  4, 14, 15, 1]
```

【例 3-39】 利用 tril 函数生成由符号矩阵 A 主对角线上的元素组成的矩阵，以及由符号矩阵 A 主对角线下的元素，组成的矩阵 A 主对角线的元素组成的矩阵，输入指令如下。

```
>> syms a b c;
>> A=[a^b c a 7;a+b b c 3;a^3 b+a 6 exp(c);2 a c 5]
>> B=sym(A)
>> tril(B,1)
>> tril(B,2)
>> tril(B,-1)
>> tril(B,-2)
>> tril(B,0)
```

输出结果如下。

```
A =
      [   a^b,      c, a,      7]
      [ a + b,      b, c,      3]
      [   a^3, a + b, 6,exp(c)]
      [     2,      a, c,      5]
B =
      [   a^b,      c, a,      7]
      [ a + b,      b, c,      3]
      [   a^3, a + b, 6,exp(c)]
      [     2,      a, c,      5]
 ans =
      [   a^b,      c, 0,      0]
      [ a + b,      b, c,      0]
      [   a^3, a + b, 6,exp(c)]
      [     2,      a, c,      5]
 ans =
      [   a^b,      c, a,      0]
      [ a + b,      b, c,      3]
      [   a^3, a + b, 6,exp(c)]
      [     2,      a, c,      5]
 ans =
      [     0,      0, 0,      0]
      [ a + b,      0, 0,      0]
      [   a^3, a + b, 0,      0]
      [     2,      a, c,      0]
  ans =
      [     0,      0, 0,      0]
      [     0,      0, 0,      0]
      [   a^3,      0, 0,      0]
      [     2,      a, 0,      0]
 ans =
      [   a^b,      0, 0,      0]
      [ a + b,      b, 0,      0]
      [   a^3, a + b, 6,      0]
      [     2,      a, c,      5]
```

6. rank 函数

在线性代数中，一个矩阵 A 的列秩是 A 的线性无关的纵列的极大数目。类似地，行秩是 A 的线性无关的横行的极大数目。矩阵的列秩和行秩总是相等的，因此，它们可以简单



地被称作矩阵 A 的秩。通常表示为 r(A)、rk(A)或 rank A。

在 MATLAB 中提供了 rank 函数指令用来计算符号矩阵的秩，其具体用法如下。

❑ **rank(A)** 返回矩阵 A 的秩。

【例 3-40】 利用 rank 指令计算四阶希尔伯特矩阵的秩。输入指令如下。

```
>> A=hilb(4)
>> A=sym(A)
>> rank(A)
```

输出结果如下。

```
A =
    1.0000    0.5000    0.3333    0.2500
    0.5000    0.3333    0.2500    0.2000
    0.3333    0.2500    0.2000    0.1667
    0.2500    0.2000    0.1667    0.1429
A =
   [   1, 1/2, 1/3, 1/4]
   [ 1/2, 1/3, 1/4, 1/5]
   [ 1/3, 1/4, 1/5, 1/6]
   [ 1/4, 1/5, 1/6, 1/7]
ans =
     4
```

【例 3-41】 利用 rank 指令计算符号矩阵的秩，输入指令如下。

```
>> syms a b c;
>> A=[b*2 c a 2;c b a 3;c^2 7 b 1;b*3 a c 8]
>> rank(A)
```

输出结果如下。

```
A =
   [ 2*b, c, a, 2]
   [   c, b, a, 3]
   [ c^2, 7, b, 1]
   [ 3*b, a, c, 8]
ans =
      4
```

7. rref 函数

矩阵的简化行阶梯式是高斯-约旦消元法解线性方程组的结果，其形式如下。

$$\begin{pmatrix} 1 & \cdots & 0 & * \\ \vdots & \ddots & \vdots & * \\ 0 & \cdots & 1 & * \end{pmatrix}$$

MATLAB 提供了 rref 函数返回符号矩阵的简化行阶梯矩阵，其具体用法如下。

❑ **R=rref(A)** 在计算过程中利用高斯-约当消元法和行主元素法，返回矩阵的简化行阶梯矩阵 R。

❑ **[R,jb]=rref(A)** 返回矩阵的简化行阶梯矩阵 R 和矢量 jb。R(1:r,jb)为 r×r 阶不确矩

阵，矩阵 A 的秩为 r=length(jb)，x(jb)为线性系统 Ax=b 的边界向量。

- **[R,jb]=rref(A,tol)**　返回矩阵的简化行阶梯 R 和矢量 jb 的要求与以上提到的相同，tol 指明了返回矩阵元素的误差。

【例 3-42】 使用 rref 函数返回符号矩阵 A 的简化行阶梯矩阵，输入指令如下。

```
>> syms a b c;
>> A=[b*2 c a 2;c b a 3;c^2 7 b 1;b*3 a c 8]
>> R=rref(A)
```

输出结果如下。

```
A =
    [ 2*b, c, a, 2]
    [   c, b, a, 3]
    [ c^2, 7, b, 1]
    [ 3*b, a, c, 8]
 R =
    [ 1, 0, 0, 0]
    [ 0, 1, 0, 0]
    [ 0, 0, 1, 0]
    [ 0, 0, 0, 1]
```

【例 3-43】 使用 rref 函数返回魔方矩阵的简化行阶梯矩阵，输入指令如下。

```
>> A=magic(4)
>> A=sym(A)
>> R=rref(A)
```

输出结果如下。

```
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
A =
    [ 16,  2,  3, 13]
    [  5, 11, 10,  8]
    [  9,  7,  6, 12]
    [  4, 14, 15,  1]
 R =
    [ 1, 0, 0,  1]
    [ 0, 1, 0,  3]
    [ 0, 0, 1, -3]
    [ 0, 0, 0,  0]
```

【例 3-44】 使用 rref 函数返回三阶希尔伯特矩阵 A 的简化行阶梯矩阵，输入指令如下。

```
>> A=hilb(3)
>> A=sym(A)
>> R=rref(A)
```

输出结果如下。

```
A =
    1.0000    0.5000    0.3333
    0.5000    0.3333    0.2500
    0.3333    0.2500    0.2000
A =
   [   1, 1/2, 1/3]
   [ 1/2, 1/3, 1/4]
   [ 1/3, 1/4, 1/5]
 R =
   [ 1, 0, 0]
   [ 0, 1, 0]
   [ 0, 0, 1]
```

【例 3-45】 使用 rref 函数返回三阶随机矩阵 A 的简化行阶梯矩阵，输入指令如下。

```
>> A=rand(3)
>> A=sym(A)
>> R=rref(A)
```

输出结果如下。

```
A =
    0.8147    0.9134    0.2785
    0.9058    0.6324    0.5469
    0.1270    0.0975    0.9575
A =
   [7338378580900475/9007199254740992, 8226958330713791/9007199254740992,
    627122237356493/2251799813685248]
   [8158648460577917/9007199254740992, 1423946432832521/2251799813685248,
    153933462881711/281474976710656]
   [1143795557080799/9007199254740992,  109820732902227/1125899906842624,
    8624454854533211/9007199254740992]
 R =
    [ 1, 0, 0]
    [ 0, 1, 0]
    [ 0, 0, 1]
```

8. null 函数

与线性系统相联系的两个子空间是值域和零空间。如果 A 为 m×n 的矩阵，其秩为 r，那么，A 的向量空间就是由 A 的列划分的线性空间，这个空间的维数是 r，也就是 A 的秩。如果 r=n，则 A 的列线性无关。A 的零空间是由满足 Ax=0 的所有向量 x 组成的线性子空间。在 MATLAB 中可以用 null 函数求得零空间的正交基，其具体用法如下。

- **N=null(A)** 计算矩阵 A 的零空间的正交基，运算依赖矩阵 A 的奇异值分解。
- **N=null(A,'r')** 计算矩阵 A 的零空间的正交基，运算依赖矩阵 A 的简化行阶梯矩阵。

【例 3-46】 使用 null 函数返回符号矩阵 A 的零空间正交基，输入指令如下。

```
>> syms a b c;
>> A=[b*2 c a 2;c b a 3;c^2 7 b 1;b*3 a c 8]
```

```
>> N=null(A)
```

输出结果如下。

```
A =
    [ 2*b,  c,  a,  2]
    [   c,  b,  a,  3]
    [ c^2,  7,  b,  1]
    [ 3*b,  a,  c,  8]
 N =
     [ empty sym ]
```

【例 3-47】 使用 null 函数返回魔方矩阵 A 的零空间正交基，输入指令如下。

```
>> A=magic(4)
>> A=sym(A)
>> N=null(A)
```

输出结果如下。

```
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
A =
    [ 16,  2,  3, 13]
    [  5, 11, 10,  8]
    [  9,  7,  6, 12]
    [  4, 14, 15,  1]
N =
    -1
    -3
    3
    1
```

9. colspace 函数

MATLAB 提供了 colspace 函数计算矩阵空间的基，其具体用法如下。

❑ **C=colspace(A)**　返回符号矩阵 A 的列空间的基。

【例 3-48】 使用 colspace 函数计算符号矩阵 A 的列空间的基，输入指令如下。

```
>> A=rand(4)
>> A=sym(A)
>> C=colspace(A)
```

输出结果如下。

```
A =
    0.8147    0.6324    0.9575    0.9572
    0.9058    0.0975    0.9649    0.4854
    0.1270    0.2785    0.1576    0.8003
    0.9134    0.5469    0.9706    0.1419
```



```
A =
   [7338378580900475/9007199254740992, 1423946432832521/2251799813685248,
    8624454854533211/9007199254740992, 8621393422876569/9007199254740992]
   [8158648460577917/9007199254740992, 1098207329O2227/1125899906842624,
    8690943295155051/9007199254740992, 4371875181445801/9007199254740992]
   [1143795557080799/9007199254740992, 627122237356493/2251799813685248,
    354913107955861/2251799813685248, 1802071410739743/2251799813685248]
   [8226958330713791/9007199254740992,  153933462881711/281474976710656,
    2185580645132801/2251799813685248, 638999261770491/4503599627370496]
C =
    [ 1, 0, 0, 0]
    [ 0, 1, 0, 0]
    [ 0, 0, 1, 0]
    [ 0, 0, 0, 1]
```

【例 3-49】 使用 colspace 函数计算魔方矩阵 A 的列空间的基，输入指令如下。

```
>> A=magic(4)
>> A=sym(A)
>> C=colspace(A)
```

输出结果如下。

```
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
     4    14    15     1
A =
    [ 16,  2,  3, 13]
    [  5, 11, 10,  8]
    [  9,  7,  6, 12]
    [  4, 14, 15,  1]
C =
    [ 1, 0,  0]
    [ 0, 1,  0]
    [ 0, 0,  1]
    [ 1, 3, -3]
```

10. transpose 函数

MATLAB 提供了 transpose 函数计算矩阵的转置，其具体用法如下。

- **T=transpose(A)** 返回符号矩阵 A 的转置。

【例 3-50】 利用 transpose 函数计算符号矩阵 A 的转置矩阵，输入指令如下。

```
>> syms a b c;
>> A=[ a^b c a 7;a+b b c 3;a^3 b+a 6 exp(c);2 a c 5]
>> T=transpose(A)
```

输出结果如下。

```
A =
    [   a^b,     c, a,      7]
```

```
    [ a + b,     b, c,       3]
    [   a^3, a + b, 6, exp(c)]
    [     2,     a, c,       5]
T =
    [ a^b, a + b,     a^3, 2]
    [   c,     b,   a + b, a]
    [   a,     c,       6, c]
    [   7,     3, exp(c), 5]
```

【例 3-51】 利用 transpose 函数计算符号矩阵 A 的共轭的转置矩阵。

这里需要说明的是 A 与 A'不同，A'代表矩阵 A 的共轭矩阵，因此，在求解转置矩阵过程中所求得也是不同的矩阵。

输入指令如下。

```
>>  syms a b c;
>>  A=[ a^b c a 7;a+b b c 3;a^3 b+a 6 exp(c);2 a c 5]
>>  T=transpose(A')
```

输出结果如下。

```
A =
    [   a^b,      c, a,      7]
    [ a + b,      b, c,      3]
    [   a^3, a + b, 6, exp(c)]
    [     2,      a, c,      5]
 T =
    [          conj(a^b),             conj(c), conj(a),           7]
    [ conj(a) + conj(b),             conj(b), conj(c),           3]
    [          conj(a)^3, conj(a) + conj(b),       6, exp(conj(c))]
    [                  2,             conj(a), conj(c),           5]
```

其中，Conj 用于计算共轭。

11. eig 函数

MATLAB 提供了 eig 函数对符号矩阵进行特征值分解，即计算矩阵的特征值和特征向量，其具体用法如下。

- ❑ **E=eig(A)** 返回由方阵 A 的特征值组成的矩阵。
- ❑ **[V,D]=eig(A)** 返回方阵 A 的特征值矩阵 D 和特证矢量矩阵 V，其中，特征值矩阵 D 是由 A 的特征值为对角线组成的对角矩阵，V、D 和 A 之间满足:AV=VD。

【例 3-52】 利用 eig 函数计算 3 阶随机矩阵的特征值和特征向量，输入指令如下。

```
>>  syms a b c;
>>  A=[a 1;b d]
>>  E=eig(A)
>>  [V,D]=eig(A)
```

输出结果如下。

```
A =
   [ a, 1]
```

```
   [ b, d]
E =
   a/2 + d/2 - (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2
   a/2 + d/2 + (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2
V =
   [ (a/2 + d/2 - (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2)/b - d/b,
   (a/2 + d/2 + (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2)/b - d/b]
   [                1,                     1]
D =
   [ a/2 + d/2 - (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2,                0]
   [ 0,                a/2 + d/2 + (a^2 - 2*a*d + d^2 + 4*b)^(1/2)/2]
```

12. jordan 函数

MATLAB 提供了 jordan 函数将矩阵变换为约当标准型，计算约当标准型也就是找一个非奇异矩阵 V，是 J=V/A*V 最接近对角矩阵，其中，V 称为转换矩阵。利用矩阵分块可以简化很多有关矩阵的证明和计算，任何仿真都可以通过相似变换变为约当标准型。jordan 函数的具体用法如下。

- **J=jordan(A)**　返回矩阵 A 的约当标准型。
- **[V,J]=jordan(A)**　返回矩阵 A 的约当标准型并且给出变换矩阵 V，满足 J=V/AV。

【例 3-53】 利用 jordan 函数计算三阶魔方矩阵的特征值和特征向量，输入指令如下。

```
>>  A=magic(3)
>>  A=sym(A)
>>  [V,J]=jordan(A)
```

1

输出结果如下。

```
 A =
     8     1     6
     3     5     7
     4     9     2
A =
     [ 8, 1, 6]
     [ 3, 5, 7]
     [ 4, 9, 2]
V =
     [ (2*6^(1/2))/5 - 7/5,   - (2*6^(1/2))/5 - 7/5, 1]
     [ 2/5 - (2*6^(1/2))/5,     (2*6^(1/2))/5 + 2/5, 1]
     [                   1,                       1, 1]
J =
     [ -2*6^(1/2),        0,  0]
     [          0, 2*6^(1/2),  0]
     [          0,        0, 15]
```

13. svd 函数

在 MATLAB 中利用 svd 函数来进行矩阵的奇异值分解，奇异值分解在矩阵分解中占有极其重要的地位。

- **[U,S,V] = svd (X)**　返回一个与 X 同大小的对角矩阵 S，两个酉矩阵 U 和 V，且满足= U*S*V'。若 A 为 m×n 阵，则 U 为 m×m 阵，V 为 n×n 阵。奇异值在 S 的对角线上，非负且按降序排列。
- **[U,S,V] = svd (X,0)**　得到一个“有效大小”的分解，只计算出矩阵 U 的前 n 列，矩阵 S 的大小为 n×n。

【例 3-54】 利用[U,S,V]=svd(A)函数对符号矩阵 A 进行奇异值的分析，输入指令如下。

```
>> A=[9 8 7;6 5 4;3 2 1]
>> digits(30)
>> [U,S,V]=svd(A)
```

输出结果如下。

```
A =
     9     8     7
     6     5     4
     3     2     1
U =
    -0.8263    0.3879    0.4082
    -0.5206   -0.2496   -0.8165
    -0.2148   -0.8872    0.4082
S =
    16.8481         0         0
          0    1.0684         0
          0         0    0.0000
V =
    -0.6651   -0.6253   -0.4082
    -0.5724    0.0757    0.8165
    -0.4797    0.7767   -0.4082
```

3.4.3　科学计算

极限、微分和积分是微积分学中的核心和基础，MATLAB 提供了强大的函数指令来对其进行计算，下面做以简单介绍。

1. 符号极限的计算

极限是高等数学的出发点和基础，高等数学的许多内容部是建立在极限理论基础上的。在 MATLAB 中提供了 limit 函数指令来对极限进行运算，其用法如下。

- **limit(fx,a)**　当变量 x 趋近于常数 a 时，计算符号函数 f(x)的极限值。
- **limit(f,a)**　相当于变量 x 趋近于 a 时，计算符号函数 f(x)的极限值。在没有指定符号函数 f(x)的自变量时，使用此函数来计算符号函数的极限，系统按 findsym 函数指示的默认变量来确定符号函数 f(x)的变量。
- **limit(f)**　在没有指定变量的目标值时，系统默认变量趋近于 0 相当于变量 x 趋近于 0，计算符号函数 f(x)的极限值，系统按 findsym 函数指示的默认变量来确定符号函数 f(x)的变量。

- **limit(f,x,a,'right')和 llimitf,x,a,'left')** 由于求极限可以从两边趋近，对于某些从左面趋近和从右面 A 趋近得到的结果是不同的，针对这种情况，函数 limit 专门提供了本语句来计算函数的左极限和右极限，'right'表示符号函数 f(x)的右极限，即变星 x 从右边趋近于 a, 'left'表示符号函数 f(x)的左极限，即变量 x 从左边趋近于 a。

【例 3-55】 计算 $\lim_{x\to\infty}\frac{3x^2-2x-1}{2x^3-x^2+5}$，输入指令如下。

```
>> syms x;
>> f=(3*x*x-2*x-1)/(7*x*x*x-5*x*x+3);
>> limit(f,x,inf)
```

输出结果如下。

```
ans =
    3/7
```

【例 3-56】 计算 $\lim_{x\to1}(2x-1)$，输入指令如下。

```
>> syms x;
>> f=2*x-1;
>> limit(f,x,1)
```

输出结果如下。

```
ans =
    1
```

【例 3-57】 计算 $\lim_{x\to0}\left(\frac{a^x+b^x+c^x}{3}\right)^{\frac{1}{x}}$，输入指令如下。

```
>> syms a b c x;
>> f=((a^x+b^x+c^x)/3)^(1/x);
>> limit(f,a)
```

输出结果如下。

```
ans =
     (a^a + b^a + c^a)^(1/a)/3^(1/a)
```

【例 3-58】 计算 $\lim_{x\to2}\frac{x^2+5}{x-3}$，输入指令如下。

```
>> syms x;
>> f=(x^2+5)/(x-3);
>> limit(f,x,2)
```

输出结果如下。

```
ans =
      -9
```

【例 3-59】 计算 $f=\lim\limits_{x\to 0}\dfrac{\tan x-\sin x}{x^3}$ 的极限，输入指令如下。

```
>> syms x;
>> f=(tan(x)-sin(x))/x^3;
>> limit(f,x,0,'left')
```

输出结果如下。

```
ans =
    1/2
```

【例 3-60】 计算【例 3-59】中函数的右极限，输入指令如下。

```
>> syms x;
>> f=(tan(x)-sin(x))/x^3;
>> limit(f,x,0,'right')
```

输出结果如下。

```
ans =
    1/2
```

【例 3-61】计算 $f=\dfrac{1}{x^3}$ 当 x 趋于 0 时的极限值，并分别求出函数的左极限和右极限，输入指令如下。

```
>> syms x;
>> f=1/x^3;
>> limit(f,x,0)
>> limit(f,x,0.'left')
>> limit(f,x,0,'right')
```

输出结果如下。

```
ans =
      NaN
ans =
      -Inf
ans =
     Inf
```

2. 符号微分的计算

MATLAB 提供了 diff 函数指令计算符号表达式的微分，具体用法如下。

- **diff(s)** 没有指定变量和导数阶数，则系统按 findsym 函数指定的默认变量对符号表达式 s 求阶导数。
- **diff(s,'v')** 以 v 为自变量，对符号表达式 s 求一阶微分。
- **diff(s,n)** 按 findsym 函数指示的默认变量对符号表达式 s 求 n 阶微分，且 n 为正整数。
- **diff(s,'v',n)** 以 v 为自变量，对符号表达式 s 求 n 阶微分，n 为正整数。

【例 3-62】 计算 $2x^3+4x^2+\cos(x)+\sin2(x)$函数关于 x 的微分，输入指令如下。

```
>> f=sym('2*x^3+4*x^2+cos(x)+sin(x)^2');
>> df=diff(f)
```

输出结果如下。

```
df =
    8*x - sin(x) + 2*cos(x)*sin(x) + 6*x^2
```

【例 3-63】 计算 $f=x-\ln(1+x)$的二阶微分，输入指令如下。

```
>> f=sym('x=ln(1+x)');
>> df=diff(f,2)
```

输出结果如下。

```
df =
0 = -1/(x + 1)^2
```

【例 3-64】 计算 $f=\sqrt{3-x}+\arctan\frac{1}{x}$关于 x 的三阶微分，输入指令如下。

```
>> f=sym('sqar(3-x)+arctan(1/x)');
>> df=diff(f,3)
```

输出结果如下。

```
df =
  14/(x^6*(1/x^2 + 1)^2) - 6/(x^4*(1/x^2 + 1)) - (D@@3)(sqar)(3 - x) - 8/(x^8*
(1/x^2 + 1)^3)
```

【例 3-65】 计算 $f=3ax^2+5bx+6$ 关于 b 的微分，输入指令如下。

```
>> f=sym('3*a*x^2+5*b*x+6');
>> df=diff(f,'b')
```

输出结果如下。

```
df =
    5*x
```

【例 3-66】 计算 $f=10x$ 关于 b 的微分，输入指令如下。

```
>> syms x
>> f=10*x*exp(-x/2);
>> df=diff(f)
```

输出结果如下。

```
df =
    10/exp(x/2) - (5*x)/exp(x/2)
```

3. 符号积分的计算

积分运算是微分运算的逆运算，MATLAB 提供了 int 函数指令计算符号表达式的积分。

该函数既可以计算定积分，也可以计算不定积分和广义积分，其具体用法如下。

- **int(s)** 没有指定积分变量和积分阶数，系统按 findsym 函数指示的默认变量对被积函数或符号表达式 s 求不定积分。
- **int(s,v)** 以 v 为自变量，计算被积函数或符号表达式 s 的不定积分。
- **int(s,v,a,b)** 计算表达式 s 的定积分，该函数是求在[a,b]区间上的定积分，a 和 b 分别是定积分的下限和上限。a 和 b 可以是两个具体的数，也可以是一个符号表达式或是无穷(inf)，当 a 和 b 有一个或两个是 inf 时，函数返回一个广义积分；当 a 和 b 中有一个符号表达式时，函数返回一个符号函数。系统按 findsym 函数指示的默认变量来确定表达式的变量，当表达式 s 是符号矩阵时，则对矩阵的各元素分别进行积分。
- **int(s,v,a,b)** 符号表达式采用符号标量 v 作为标量，求 v 从 a 变到 b 时，符号表达式 s 的定积分值，a 和 b 的规定同上。

【例 3-67】 计算 f=tan3(x)关于 x 的不定积分，输入指令如下。

```
>> f=sym('tan(x)^3');
>> int(f)
```

输出结果如下。

```
ans =
     log(cos(x)) - (cos(x)^2 - 1)/(2*cos(x)^2)
```

【例 3-68】 计算 f=x3+acos(x)+bsin2(x)关于 b 的不定积分，输入指令如下。

```
>> f=sym('x^3+a*cos(x)+b*sin(x)^2');
>> int(f,'b')
```

输出结果如下。

```
ans =
     b* (a*cos(x) + x^3) - b^2* (cos(x)^2/2 - 1/2)
```

【例 3-69】 计算定积分 $\int_{\frac{\pi}{4}}^{\frac{\pi}{3}} \frac{x}{\sin^2 x} dx$，输入指令如下。

```
>> syms x;
>> f=x/sin(x)^2;
>> int(f,x,pi/4,pi/3)
```

输出结果如下。

```
ans =
     pi/4 + log(6^(1/2)/2) - (pi*3^(1/2))/9
```

【例 3-70】 计算定积分 x^2+16y^2，输入指令如下。

```
>> syms x y;
>> f=x^2+16*y^2;
>> int(f,x)
```

```
>> int(f,y)
>> a=5;b=10
>> int(f,x,a,b)
```

输出结果如下。

```
ans =
      x^3/3 + 16*x*y^2
ans =
      x^2*y + (16*y^3)/3
ans =
     80*y^2 + 875/3
```

4. 级数求和的计算

MATLAB 提供了 symsum 函数指令用于计算符号表达式的和，其具体用法如下。

- **r=symsum(s)** 自变量是由 findsym 函数所确定的符号变量，默认自变量为 k，计算表达式 s 从 0 到 k-1 的和。
- **r=symsum(s,v)** 计算表达式 s 从 0 到 v-1 的和。
- **r=symsum(s,a,b)** 计算表达式 s 默认变量从 a 到 b 的和。
- **r=symsum(s,v,a,b)** 计算表达式 s 变量 v 从 a 到 b 的和。

【例 3-71】 利用 symsum 函数计算符号表达式的和，输入指令如下。

```
>>  syms x k;
>>  symsum(x+3*x^4)
>>  symsum(1/x^k,k,0,inf)
>>  symsum(1/x^k,k,1,inf)
```

输出结果如下。

```
ans =
     (3*x^5)/5 - (3*x^4)/2 + x^3 + x^2/2 - (3*x)/5
ans =
     x/(-1+x)
ans =
     x/(-1+x)
```

【例 3-72】 对$\sum(a*n^2+b*n)$其中 n 从 1 变到 20，输入指令如下。

```
>> syms a b n;
>> f=a*n^2+b*n;
>> symsum(f,n,1,20)
```

输出结果如下。

```
ans =
    2870*a + 210*b
```

5. taylor 级数的计算

MATLAB 提供了 taylor 函数用来计算符号表达式的泰勒级数展开式，其具体用法如下。

- **r=taylor(f)**　f是符号表达式，自变量是由 findsym 函数所确定的符号变量，该函数将返回 f 在变量等于 0 处进行 5 阶泰勒展开时的展开式。
- **r=taylor(f,n,v)**　符号表达式 f 以符号标量 v 作为自变量，返回 f 的 n-1 阶麦克劳林级数(即在 v=0 处进行泰勒展开)展开式。
- **r=taylor(f,n,v,a)**　返同符号表达式 f 在 v=a 处进行 n-1 阶泰勒展开的展开式。

【例 3-73】　使用 taylor 函数指令计算符号表达式的泰勒级数展开式，输入指令如下。

```
>> syms x;
>> f=f*x^3;
>> T=taylor(f)
```

输出结果如下。

```
T =
    x^4/3 + x^2
```

【例 3-74】　使用 taylor 函数指令计算符号表达式的泰勒级数展开式，输入指令如下。

```
>> syms x a;
>> f=x^a;
>> T=taylor(f,4,a)
```

输出结果如下。

```
T =
    (a^3*log(x)^3)/6 + (a^2*log(x)^2)/2 + a*log(x) + 1
```

3.5　符号表达式积分变换

积分变换是工程设计和计算常用的工具，常见的积分变换有傅里叶（Fourier）变换、拉普拉斯（Laplace）变化和 Z 变换。

3.5.1　傅里叶变换及其反变换

傅里叶变换是一种分析信号的方法，它可分析信号的成分，也可用这些成分合成信号。许多波形可作为信号的成分，如正弦波、方波、锯齿波等，傅里叶变换用正弦波作为信号的成分。

$f(x)$是关于 x 的函数，如果 x 满足狄里赫莱条件，具有有限个间断点；具有有限个极值点；绝对可积。则有如下公式成立：$F(\omega)=\int_{-\infty}^{+\infty} f(x)e^{-j\omega x}\mathrm{d}x$。

时域中的 $f(x)$ 和频域中的 $F(\omega)$ 的傅里叶反变换存在如下关系：

$$f(x)=\frac{1}{\pi}\int_{-\infty}^{\infty} F(\omega)e^{-j\omega x}\mathrm{d}\omega$$

1．傅里叶变换

在 Matlab 中进行傅里叶变换的函数指令如下。

- **fourier(f)** 默认函数 f 的自变量是 x，对默认变量计算傅里叶变换时，并且默认输出结果 F 是变量 ω 的函数，记为 $F(\omega)=\int_{-\infty}^{+\infty} f(x)e^{-j\omega x}\mathrm{d}x$。
- **fourier(f,v)** 在默认函数 f 的自变量是 u、制定参数是 v，求函数 f 的傅里叶变换，记为 $F(v)=\int_{-\infty}^{+\infty} f(x)e^{-jvx}\mathrm{d}x$。
- **fourier(f,u,v)** 制定函数 f 的自变量是 u，指定参数变为 v，对函数 f 进行傅里叶变换，记为 $F(v)=\int_{-\infty}^{+\infty} f(u)e^{-jvu}\mathrm{d}u$。

【例 3-75】 分别用默认 fourier 函数和指定参数 fourier(f,v)函数计算 f(x)的傅里叶变换，输入指令如下。

```
>> syms x w u v;
>> f=sin(x)-cos(x)+1;
>> fourier(f)
>> fourier(f,v)
```

输出结果如下。

```
ans =
      2*pi*dirac(w) - pi* (dirac(w - 1) + dirac(w + 1)) - pi* (dirac(w - 1)
      - dirac(w + 1)) *i
 ans =
      2*pi*dirac(v) - pi* (dirac(v - 1) + dirac(v + 1)) - pi* (dirac(v - 1)
       - dirac(v + 1)) *i
```

【例 3-76】 使用 fourier(f,u,v)函数在指定自变量和变换参数的情况下计算 f(x)的傅里叶变换，输入指令如下。

```
>> syms t u v;
>> f=exp(-1/3* (t+u)^2);
>> fourier(f,t,v)
```

输出结果如下。

```
ans =
    (3^(1/2) *pi^(1/2))/exp((3* (- v + (2*u*i)/3)^2)/4 + u^2/3)
```

2. Fourier 反变换

- **ifourier(F)** 在系统默认自变量和变换参数的情况下，计算函数的傅里叶反变换，记为 $f(x)=\frac{1}{2\pi}\int_{-\infty}^{+\infty} F(\omega)e^{-j\omega x}\mathrm{d}\omega$。
- **ifourier(F,v)** 在系统默认自变量，并指定变换参数是 v 的情况下，计算函数的傅里叶反变换，记为 $f(v)=\frac{1}{2\pi}\int_{-\infty}^{+\infty} F(\omega)e^{-j\omega v}\mathrm{d}\omega$。
- **ifourier(F,w,v)** 在系统的自变量为 w，并制定变换参数是 v 的情况下，计算函数的傅里叶反变换，记为 $f(v)=\frac{1}{2\pi}\int_{-\infty}^{+\infty} F(w)e^{-j\omega v}\mathrm{d}w$。

【例 3-77】 使用函数 f= ifourier(F)在系统默认自变量和变换参数的情况下，计算函数的博里叶反变换，输入指令如下。

```
>> syms x w u v;
>> f=sin(x)-cos(x)+1;
>> ifourier(f)
```

输出结果如下。

```
ans =
    (2*pi*dirac(t) - pi* (dirac(t - 1) + dirac(t + 1)) + pi* (dirac(t - 1)
    - dirac(t + 1)) *i)/(2*pi)
```

【例 3-78】 使用 ifourier(f,v)函数在指定自变量和变换参数的情况下计算 f(x)的傅里叶反变换，输入指令如下。

```
>> syms x y;
>> f=exp(-(x+y)^2/3);
>> ifourier(f,v)
```

输出结果如下。

```
ans =
    3^(1/2)/(2*pi^(1/2)*exp((3* (v + (2*y*i)/3)^2)/4 + y^2/3))
```

3.5.2 拉普拉斯变换及其反变换

拉普拉斯变换是工程数学中常用的一种积分变换，又名拉氏转换。拉氏变换是一个线性变换，可将一个有引数实数 $t(t\geqslant 0)$的函数转换为一个引数为复数 s 的函数。

如果函数 $f(t)$ 在区间 $[0,+\infty)$ 上有定义，并且积分 $\int_0^{+\infty} f(t)e^{-st}\mathrm{d}t$ 在 s 的某一区域内收敛，则由这个积分确定函数 $F(s)$，即 $F(s)=\int_0^{+\infty} f(t)e^{-st}\mathrm{d}t$，此式称为函数 $f(t)$ 的拉普拉斯变换式，记为 $L[f(t)]=F(s)$。

拉普拉斯反变换定义为：$F(t)=\dfrac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty} L(s)e^{st}\mathrm{d}t$，其中 c 为使函数 $L(s)$的所有奇点位于直线 $s=c$ 左边的实数，Laplace 反变换记为：$F(t)=L^{-1}[L(s)]$。

1．拉普拉斯变换

在 MATLAB 中提供了如下函数来进行拉普拉斯变换。

- **laplace(F)** 在默认自变量(自变量默认为 x)和参变量(参变量默认为 s)的情况下，计算符号函数的 Laplace 变换，记为 $L(s)=\int_0^{+\infty} F(x)e^{-sx}\mathrm{d}x$。
- **laplace(F,z)** 在默认自变量(自变量默认为 x)，并指定参变量为 z 的情况下，计算函数的拉普拉斯变换，记为 $L(z)=\int_0^{+\infty} F(x)e^{-zx}\mathrm{d}x$。
- **laplace(F,w,z)** 在指定自变量为 w，并指定参变量为 z 的情况下，计算函数的拉普拉斯变换，记为：$L(z)=\int_0^{+\infty} F(w)e^{-zw}\mathrm{d}w$。

【例 3-79】 使用函数 laplace(F)在默认自变量和参变量的情况下，计算符号函数的拉普拉斯变换，输入指令如下。

```
>> syms x;
>> f=2*sin(3*x);
>> laplace(f)
```

输出结果如下。

```
ans =
    6/(s^2 + 9)
```

【例 3-80】 使用函数 laplace(F,w,z)和 f(F,z)分别计算在指定自变量为 w，并指定参变量为 z 的情况下，计算符号函数的拉普拉斯变换，输入指令如下。

```
>> syms w z x;
>> f=1+log(x+2);
>> l=laplace(f,w,z)
>> l=laplace(f,z)
```

输出结果如下。

```
l =
   (log(x + 2) + 1)/z
l =
   1/z + laplace(log(x + 2), x, z)
```

2．进行 laplace 函数范变化的指令如下

- **ilaplace(L)** 在默认自变量(自变量默认为 s)和参变量(参变量默认为 t)的情况下，计算函数 L(s)的拉普拉斯反变换，记为 $L^{-1}[L(s)]=F(t)=\dfrac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty}L(s)e^{st}\mathrm{d}s$，其中，c 为使函数 L(s)的所有奇点位于直线 s=c 左边的实数。
- **ilaplace(L,v)** 在默认自变量(自变量默认为 s)并指定参变量 v 的情况下，计算函数 L(s)的拉普拉斯反变换，记为 $L^{-1}[L(s)]=F(v)=\dfrac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty}L(s)e^{sv}\mathrm{d}s$，其中，c 为使用函数 L(s)的所有奇点位于直线 s=c 左边的实数。
- **ilaplace(L,v,x)** 在指定自变量为 x 并指定参变量为 v 的情况下，计算函数 L(s)的拉普拉斯反变换，记为 $L^{-1}[L(x)]=F(v)=\dfrac{1}{2\pi i}\int_{c-i\infty}^{c+i\infty}L(x)e^{sv}\mathrm{d}x$，其中 c 为使用函数 L(s)的所有奇点位于直线 s=c 左边的实数。

【例 3-81】 使用函数 ilaplace(L)在默认自变量和参变量的情况下，计算函数 L(s)的拉普拉斯反变换，输入指令如下。

```
>> syms x;
>> f=cos(x-2);
>> L=ilaplace(f)
```

输出结果如下。

```
L =
    ilaplace(cos(x - 2), x, t)
```

【例 3-82】 使用函数 ilaplace(L,v)和 ilaplace(L,v,x)计算函数的拉普拉斯反变换，输入指令如下。

```
>> syms v x w;
>> L=(v^3+w^2+3);
>> ilaplace(L,v)
```

输出结果如下。

```
ans =
     3*dirac(v) + v^3*dirac(v) + dirac(v, 2)
```

3.5.3 Z 变换及其反变换

Z 变换的算法：当 t<0 时，$f^*(t)=f(t)=0$；当 t≥0 时，$f^*(t)=\sum_{k=0}^{+\infty}f(kT)\sigma(t-kT)$。

对该式进行拉普拉斯变换，得到 $F^*(s)=\sum_{k=0}^{+\infty}f(kT)e^{-skT}$，此时令 $z=e^{sT}$，于是函数变为 $F^*(z)=\sum_{k=0}^{+\infty}f(kT)z^{-k}$，在函数 $F^*(z)$ 收敛的情况下，称 $F^*(z)$ 是 $f^*(t)$ 的 Z 变换，Z 变换记为 $F(z)=\sum_{k=0}^{\infty}f(n)z^{-n}$。

1．Z 变换

在 MATLAB 中进行 Z 变换的函数如下。

- ❑ **ztrans(f)** 在默认自变量(默认自变量为 n)和参变量(参变量默认为 z)的情况下，计算符号函数的 Z 变换，记为 $F(z)=\sum_{n=0}^{+\infty}f(n)z^{-n}$。
- ❑ **ztrans(f,v)** 在默认自变量(默认自变量为 n)并制定参变量为 v 的情况下，计算符号函数的 Z 变换，记为 $F(v)=\sum_{n=0}^{+\infty}f(n)v^{-n}$。
- ❑ **ztrans(f,k,v)** 在制定自变量为 k，并制定参变量为 v 的情况下，计算符号函数的 Z 变换，记为 $F(z)=\sum_{n=0}^{+\infty}f(k)v^{-k}$。

【例 3-83】使用 ztrans(f)函数指令在默认自变量和参变量的情况下对函数进行 Z 变换，输入指令如下。

```
>> syms x;
>> f=cos(4*x);
>> ztrans(f)
```

输出结果如下。

```
ans =
     (z* (z - cos(4)))/(z^2 - 2*cos(4) *z + 1)
```

【例 3-84】 使用 ztrans(f,v)函数与 ztrans(f,k,v)分别对函数进行 Z 变化，输入指令如下。

```
>> syms k m v;
>> f=tan(3*k*m);
>> ztrans(f,v)
```

输出结果如下。

```
ans =
    ztrans(tan(3*k*m), m, v)
```

2. Z 变换的反变换

Z 反变换记为：$f(n) = Z^{-1}(F(z))$。

MATLAB 提供了 iztrans 函数指令来实现 Z 反变换，其调用方法如下。

- ❑ **iztrans(F)** 在默认自变量(默认自变量为 n)和参变量(参变量为 z)的情况下，对函数进行 Z 反变换，记为 $f(n) = \frac{1}{2\pi i}\int_{|z|=R} F(z)z^{n-1}\mathrm{d}z, n = 1,2,3,\ldots$
- ❑ **itrans(F,v)** 在默认自变量(默认自变量为 n)并指定参变量为 v 的情况下，对函数进行 Z 反变换，记为 $f(n) = \frac{1}{2\pi i}\int_{|v|=R} F(v)v^{n-1}\mathrm{d}v, v = 1,2,3,\ldots$
- ❑ **itrans(F,w,v)** 在指定自变量为 w 并指定参变量为 v 的情况下，对函数进行 Z 反变换，记为 $f(w) = \frac{1}{2\pi i}\int_{|v|=R} F(v)v^{w-1}\mathrm{d}v, v = 1,2,3,\ldots$

【例 3-85】使用 iztrans(F)函数指令在默认自变量和参变量的情况下对函数进行 Z 反变换，输入指令如下。

```
>> syms w,v;
>> f=w* (w-3) * (w/2+f*w-2);
>> iztrans(f,v)
```

输出结果如下。

```
ans =
    3*iztrans(x^2, x, n) - iztrans(x^3, x, n)
```

【例 3-86】 使用函数指令 iztrans(F,v)和 iztrans(F,w,v)分别计算函数的 Z 反变换，输入指令如下。

```
>> syms w,v;
>> f=w* (w-3) * (w/2+f*w-2);
>> iztrans(f,v)
```

输出结果如下。

```
ans =
     (w^3*kroneckerDelta(v, 0))/2 - (7*w^2*kroneckerDelta(v, 0))/2 +
     (3*w^2 - w^3)*iztrans(x^3, x, v) - (9*w^2 - 3*w^3)*iztrans(x^2, x, v)
      + 6*w*kroneckerDelta(v, 0)
```

3.6 符号函数的图形绘制

图形是解决数学问题的必要途径，MATLAB 除了为解决代数方程提供了支持外还对函数图像的绘制提供了强大的支持。本节对简单的符号函数绘制加以简单介绍，详细的图形绘制在第 4 章进行系统学习。

3.6.1 符号函数的曲线绘制

MATLAB 提供了 ezplotl 函数和 ezplot3 函数用于绘制符号函数的二维曲线和三维曲线。

1．二维曲线的绘制

MATLAB 提供了 ezplot 函数来绘制符号函数的二维曲线，此函数可以绘制显函数图形、隐函数图形和参数方程的图形，具体用法如下。

- **ezplot(f)**　绘制显函数 f 在区间[–2,2]的二维曲线；绘制参数方程 $x=x(t)$，$y=y(t)$在区间 $0<t<2$ 的曲线。
- **ezplot(f,[min,max])**、**ezplot(f,[xmin,xmax,ymin,ymax,])**和 **ezplot(x,y,[min,tmax])**　第一种用法是绘制显函数 f 在指定区间[min,max]的二维曲线；第二种用法是绘制隐函数 f 在指定区间 xmin<x<xmax、ymin<y<ymax 的曲线；第三种用法是绘制参数方程 x=x(t)、y=y(t)在区间 tmin<t<tmax 的曲线。

【例 3-87】 使用 ezplot(f)函数指令绘制显函数的二维曲线，输入指令如下。

```
>> syms x;
>>f=sin(x);
>> ezplot(f);
>> grid;
>> title('sin(x)');
```

输出图形如图 3-1 所示。

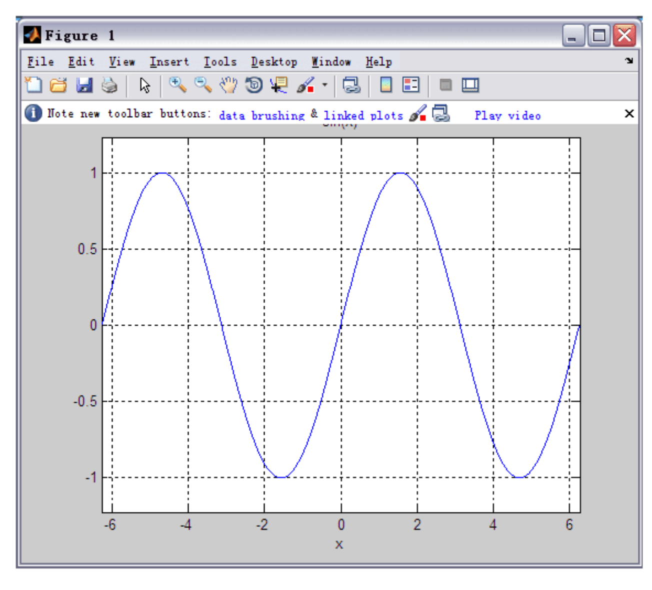


图 3-1　ezplot(f)函数二维曲线

【例 3-88】 绘制函数 $y = x^3 - x^2 - x + 1$ 的二维曲线，输入指令如下。

```
>> syms x;
>>f=x^3-X^2-x+1;
>> ezplot(f);
>> grid;
>> title('exp');
```

输出图形如图 3-2 所示。

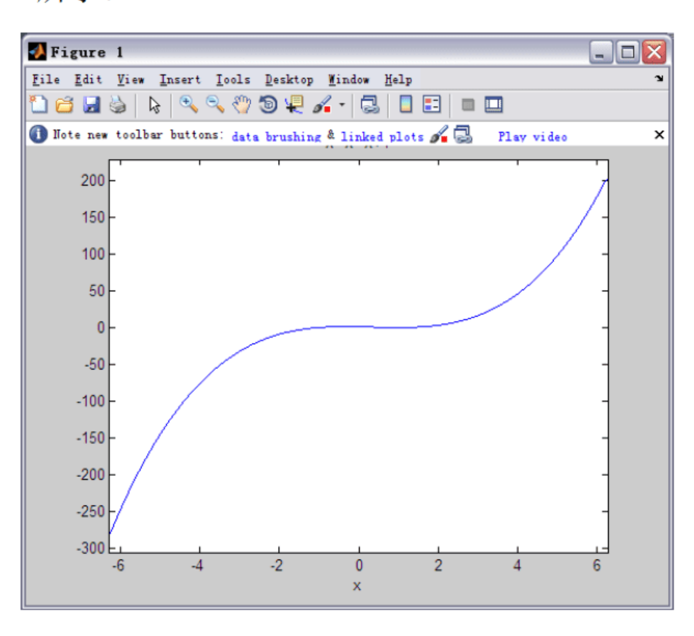


图 3-2　二维曲线

2. 三维曲线的绘制

ezplot3 函数用于绘制符号函数的三维曲线，具体用法如下。

- ❑ **ezplot3(x,y,z)**　绘制参数方程 $x=x(t)$、$y=y(t)$、$z=z(t)$在默认区间 $0<t<2$ 的三维曲线。
- ❑ **ezplot3(x,y,z[t,min,tmaxJ)**　绘制参数方程 $x=x(t)$, $y=y(t)$, $z=z(t)$在区间 tmin<t<tmax 的三维曲线。ezplot3(...'animate')，生成空间曲线的动态轨迹。

【例 3-89】 绘制函数的三维曲线，输入指令如下。

```
>> syms t;
>>x=cos(t);
>> y=tan(t);
>> z=t;
>> ezplot3(x,y,z)
```

输出图形如图 3-3 所示。

3.6.2　符号函数等值线的绘制

MATLAB 提供了 ezcontour 函数和 ezcontourfd 函数用于绘制符号函数的等值线，两个函数的使用方法类似，区别在于 ezcontourfd 函数绘制带有填充区域的等值线。ezcontour 函数的具体用法如下。

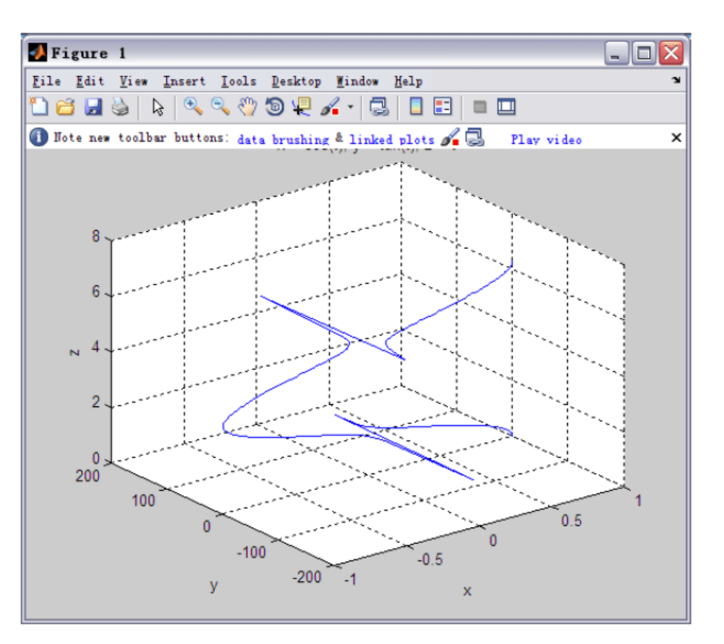


图 3-3　三维曲线

- **ezcontour(f)**　绘制二元函数 f(x,y)在默认区域的等值线。
- **ezcontour(f,domain)**　绘制二元函数 f(x,y)在指定区域的等值线。
- **ezcontour(...,n)**　绘制等值线图，并指定等值线的条数。

【例 3-90】 使用 ezcontour 函数绘制符号函数的等值线，输入指令如下。

```
>> syms x;
>>f=x^3-x^2-x+1;
>> ezcontour(f);
```

输出图形如图 3-4 所示。

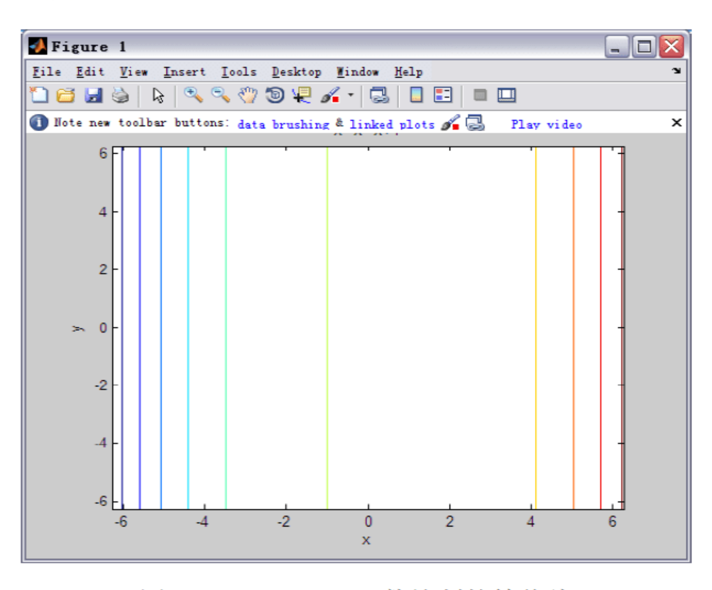


图 3-4　ezcontour 函数绘制的等值线

【例 3-91】 使用 ezcontourf 函数绘制符号函数的等值线，输入指令如下。

```
>> syms x y;
>>f=2* (1+y)^2*exp(-(x^3)-(x+1)^2);
>> ezcontourf(f);
```

输出图形如图 3-5 所示。

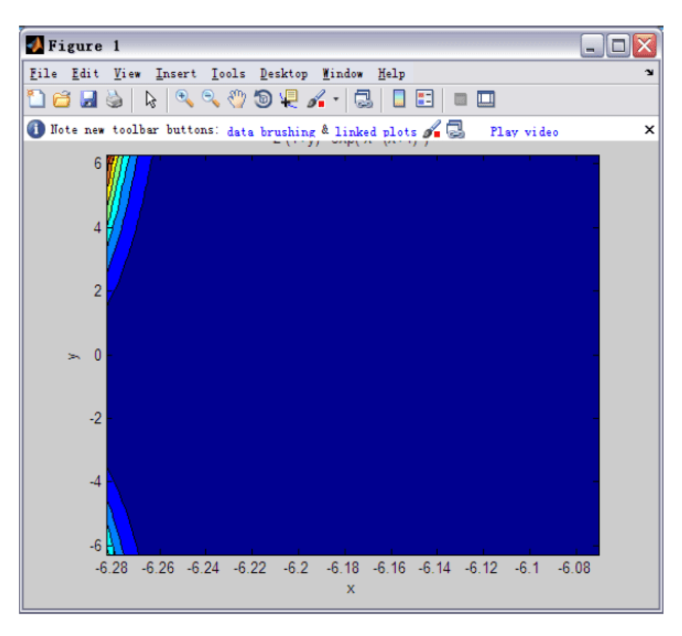


图 3-5 ezcontourf 函数绘制的等值线

3.6.3 符号函数曲面图及表面图的绘制

MATLAB 提供了 ezmesh 函数和 ezmeshc 函数用于绘制符号函数的三维曲面图，以及 ezsurf 函数和 ezsurfc 函数用于绘制符号函数的三维表面图。

ezmesh 函数和 ezmeshc 函数的区别在于 ezmeshc 函数在绘制三维曲面图的同时绘制等值线，ezsurf 函数和 ezsurfc 函数的区别在于 ezsurfc 函数在绘制三维表面图的同时绘制等值线。

1. ezmesh 函数和 ezsurf 函数

ezmesh 函数用于绘制三维曲面图，ezsurf 函数绘制三维表面图。ezmesh 的具体用法如下。

- **ezmesh(f)** 绘制 $f(x,y)$的图像。
- **ezmesh(f,domain)** 在指定区域绘制 $f(x,y)$的图像。
- **ezmesh(x,y,z)** 在默认区域绘制三维参数方程的图像。
- **ezmesh(x,y,z, [smin,smax,tmin,tmax])或 ezmesh(x,y,z,[min,max])** 在指定区域绘制维参数方程的图像。

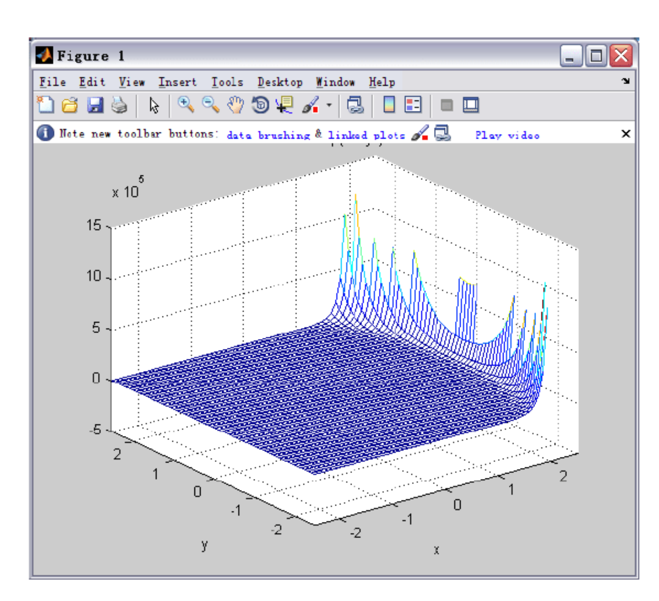
【例 3-92】 利用 ezmesh 函数和 ezsurf 函数绘制三维曲面图和三维表面图，输入指令如下。

```
>> syms x y;
>>ezmesh(x*exp(x^3+y^2),[-2.5,2.5]);
>> ezsurf(x*exp(x^3+y^2),[-2.5,2.5]);
```

输出图形如图 3-6 所示。

2. ezmeshc 函数和 ezsurfc 函数

MATLAB 提供了 ezmeshc 函数用于绘制带等值线的三维曲面图，ezsurfc 函数绘制带等值线的三维表面图，两个函数使用的方法类似。ezmeshc 函数的具体用法如下。



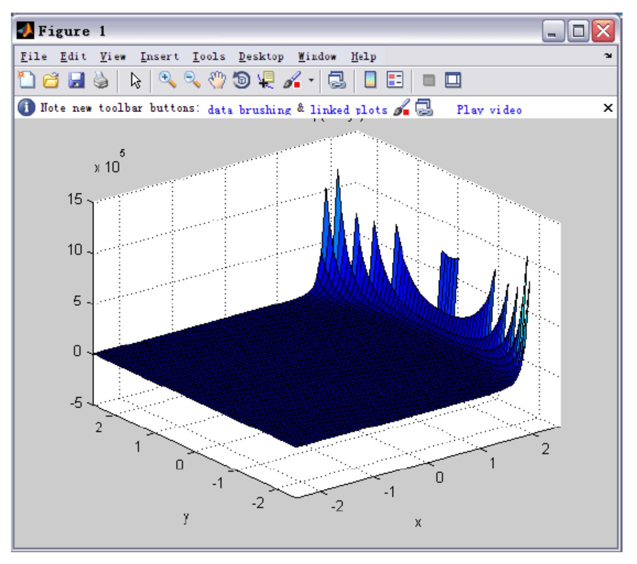


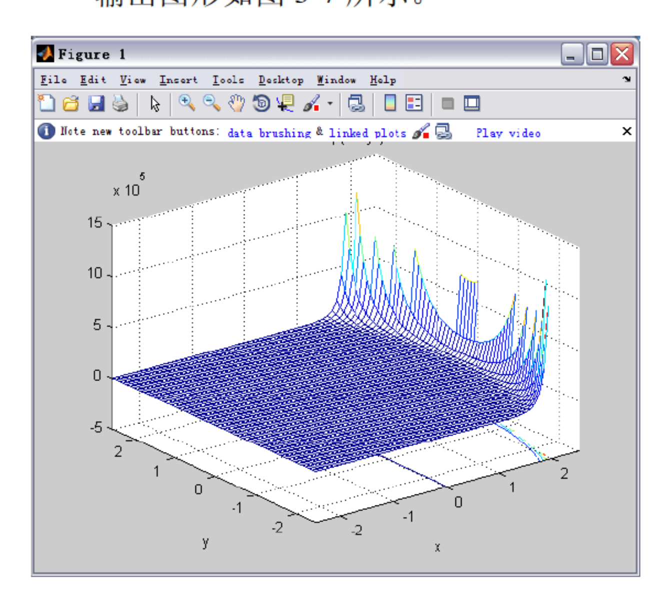
图 3-6　三维曲面图和三维表面图

- ❑ **ezmeshc(f)**　在默认区域$-2\pi<x<2\pi$，$-2\pi<y<2\pi$绘制二元函数 f(x,y)的图像。
- ❑ **ezmeshc(f,domain)**　在指定区域绘制二元函数 f(x,y)的图像。
- ❑ **ezmeshc(x,y,z)**　在默认区域$-2\pi<s<2\pi$，$-2\pi<t<2\pi$绘制三维参数方程 x=x(s,t), y=y(s,t),z=z(s,t)的图像。
- ❑ **ezmeshc(x,y,z,[smin,smax,tmin,tmax])或 ezmeshc(x,y,z,[min,max])**　在指定区域绘制三维参数方程的图像。

【例 3-93】使用 ezmeshc 函数和 ezsurfc 函数绘制带等值线的三维曲面图和三维表面图，输入指令如下。

```
>> syms x y;
>>ezmeshc(x*exp(x^3+y^2),[-2.5,2.5]);
>> syms x y;
>> ezsurfc(x*exp(x^3+y^2),[-2.5,2.5]);
```

输出图形如图 3-7 所示。



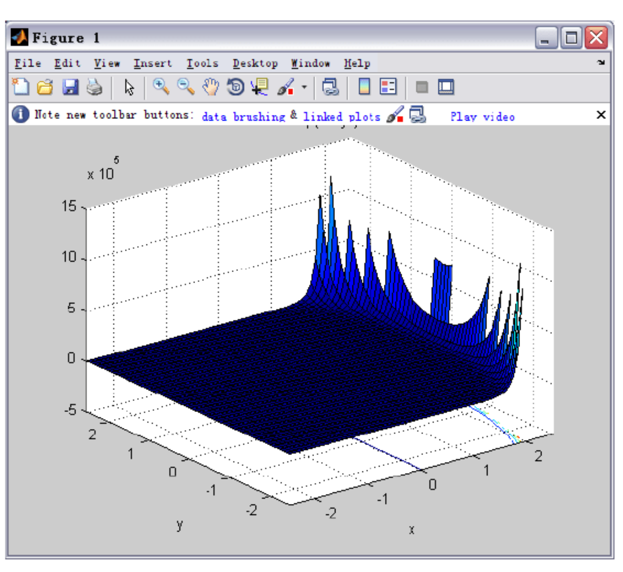


图 3-7　三维曲面图和三维表面图

3.7 符号方程的求解

在学习代数的过程中，我们一直在探索关于方程求解的各种理论与方法，从最基础的消元法到高斯迭代，方程成为我们在数学计算中必不可少的步骤，其重要性不言而喻。MATLAB 为方程的求解提供了更便利的方法。

3.7.1 代数方程的求解

代数方程分为线性方程、非线性方程以及超越方程。对于代数方程，MATLAB 提供了一种对方程的求解指令 solve，其格式如下。

- ❑ **g=solve(eq)** 其中 eq 可以是符号表达式或不带符号的字符串，该函数用于求解方程 eq=0，其自变量采用默认变量，也可以通过 findsym 函数来确定。
- ❑ **g=(eq,var)** 求解方程 eq=0，其自变量参数由 var 来指定，其中 eq 和上一种调用方式相同，返回值 g 是由方程所有解构成的列向量。

【例 3-94】 使用 solve 函数解代数方程 2*ax*2+3*bx*-*c*=0，输入指令如下。

```
>> syms x a b c;
>>x=solve('2*a*x^2+3*b*x-c');
```

输出结果如下。

```
x=
   -1/4* (3*b-(9*b^2+8*a*c)^(1/2))/a
   -1/4* (3*b-(9*b^2+8*a*c)^(1/2))/a
```

【例 3-95】 使用 solve 函数解代数方程 4*ax*2+*bx*+*c*=0，其中 b 为自变量，输入指令如下。

```
>> syms x a b c;
>>x=solve('4*a*x^2+b*x+c'.'b');
```

输出结果如下。

```
x=
   -(4*a*x^2+c)/x
```

solve 求解指令同样适用于求解代数方程组，指令格式如下。

- ❑ **g=(eq1,eq2,...eqn)** 求解由符号表达式或不带符号的字符串 eq1,eq2,...eqn 组成的方程组，其中，自变量为整个方程组的默认变量，即将函数 findsym 作用于整个方程组时返回的变量。
- ❑ **g=(eq1,eq2,...eqn,var1,var2...,varn)** 求解由符号表达式或不带符号的字符串 eq1,eq2,...eqn 组成的方程组，其自变量由输入参数 var1,var2...,varn 指定。

【例 3-96】 使用 solve 函数解代数方程组 $\begin{cases} x+y=1 \\ x+3y=5 \end{cases}$，输入指令如下。

```
>> syms x y;
>>S=solve('x+y=1', 'x+3*y=5')
>> disp('S.x'),disp(S.x),disp('S.y'),disp(S.y)
```

输出结果如下。

```
S=
   x :  [1x1 sym]
   y :  [1x1 sym]
   S.x
   -1
   S.y
   2
```

【例 3-97】 使用 solve 函数解代数方程组 $\begin{cases} x+2y=3 \\ 3x+y=8 \end{cases}$，输入指令如下。

```
>> syms x y;
>>S=solve('x+2*y=3', '3*x+y=8')
>> disp('.x'),disp(S.x),disp('S.y'),disp(S.y)
```

输出结果如下。

```
S=
   x :  [1x1 sym]
   y :  [1x1 sym]
   S.x
   1/5
   S.y
   13/5
```

3.7.2 微分方程求解

从数值计算的角度来看，微分方程要比代数方程复杂困难得多，这时候不妨通过符号计算指令来进行求解。对于符号计算来说，无论是初值问题还是边值问题，其求解微分方程的指令行事都相同，而且比较简单。但是存在的问题为符号计算可能会花费较多的计算机资源，可能得不到简单的解析解或封闭形式的解，甚至无法求解，所以没有万能的求解微分方程的一般解法。既然没有万能的微分方程一般解法，那么，求解微分方程的符号法和数值法就有很好的互补作用。

对于微分方程，MATLAB 提供了 dsolve 函数指令来对其进行求解，其函数的调用格式如下。

- **r=dsolve('eq1,eq2,... ', 'cond1,cond2,... ', 'v')** 计算由 eq1,eq2,...指定的常微分方程的符号解。常微分方程以变量 v 作为自变量，参数 cond1,cond2,...用于指定方程的边界条件或者初始条件，如果不指定 v，将默认 t 为自变量。

- **r=('eq1', 'eq2',... 'cond1','cond2', 'v')** 计算由eq1,eq2,...指定的常微分方程的符号解，这些常微分方程都以 v 作为自变量，这些单独输入的方程的最大允许个数为12，其他参数调用方式同上。

在方程中，用 D 来表示一次微分，D2、D3 分别表示二次、三次微分，以此类推。例如符号 $D2y$ 来表示 $\frac{D^2y}{Dt^2}$ 函数 dsolve 把 D 后面的字符当做因变量，并默认所有这些变量对 t 进行求导。

微分方程的初始条件或边界条件都以变量 v 作为自变量，其形式为 y(a)=b 或 Dy(a)=b，其中 y 是微分方程的因变量，a 和 b 是常数。如果指定的出事条件和边界条件比方程中的因变量个数少，那么所得的解中将包含积分常数 C1、C2 等。

函数 dsolve 的输出结果同 solve 相似，既可以用和因变量个数相同的输出参数分别接收每个变量的解，也可以把方程的解写入一个结构数组中。

【例 3-98】 使用 dsolve 函数指令求解微分方程 $\frac{dx}{dt}=-ax$，$\frac{d^2x}{dt^2}=\sin(t)$，$\left(\frac{dy}{ds}\right)^2+y^2=1$，输入指令如下。

```
>> dsolve('Dx=-a*x');
>>dsolve('D2x/d*t^2=sin(t)')
>> dsolve('(Dy)^2/+y^2=1','s')
```

输出结果如下。

```
ans =
     C2/exp(a*t)
ans =
     C5 - d*sinint(t) + (C4*t^3 + d*t^3*cosint(t) - d*t^2*sin(t))/t^2
ans =
    C7*exp(s)
 C9/exp(s)
```

【例 3-99】 使用 dsolve 函数指令在指定初值的情况下求解微分方程 $\frac{d^2y}{dt^2}=-a^2y$，输入指令如下。

```
>> dsolve('D2y=-a^2*y', 'y(0)=1', 'Dy(pi/2)=0');        %限定初值
```

输出结果如下。

```
ans =
    sin(1/2*a*pi)/cos(1/2*a*pi)*sin(a*t)+cos(a*t)
```

【例 3-100】 使用 dsolve 函数指令在计算微分方程 $\frac{dy}{dx}+3xy=xe^{-x^2}$ 的通解，输入指令如下。

```
>> dsolve('Dy+3*x*y=x*exp(-x^2)');
```

输出结果如下。

```
ans =
    (1/3*exp(-x* (x-3*t))+C1) *exp(-3*x*t)
```

由于系统默认的自变量是 t，显然系统把 x 当作常数，把 y 当做 t 的函数求解，故输入

```
>> dsolve('Dy+3*x*y=x*exp(-x^2)','x');
```

输出结果如下所示。

```
ans =
    exp(-x^2)+exp(-3/2*x^2) *C1
```

与代数方程组的求解相同，dsolve 同样也可以用于解微分方程组，函数指令的调用如下。

r=dsolve('eq1,eq2,... ', 'cond1,cond2,... ', 'v')，计算由 eq1,eq2,...指定的常微分方程组的解。v 作为自变量，参数 cond1,cond2,...用于指定方程的边界条件或初始条件。

【例 3-101】 使用 dsolve 函数指令求解微分方程组，输入指令如下。

```
>>syms x y;
>>S=dsolve('Dx=-y','Dy=x','x(0)=0','y(0)=1');
>> S.x          %查看 x 的值
>> S.y          %查看 y 的值
```

输出结果如下所示。

```
ans =
    -sin(t)
ans =
    cos(t)
```

3.7.3 复合方程的求解

了解了代数求解之后不禁会产生疑问，复合方程怎样求解呢？是否是多次代数方程求解的叠加？针对复合方程，MATLAB 提供了 compose 函数指令来求解复合方程，其调用格式如下。

- **compose(f,g)** 返回函数 $f(g(y))$，其中 x 是 f 的默认变量，即 $f=f(x)$，y 是 g 的默认变量，即 $g=g(y)$。
- **Compose(f,g,z)** 返回函数 $f(g(z))$，自变量指定为 z。
- **compose(f,g,x,z)** 返回函数 $f(g(z))$，指定 f 的自变量是 x。
- **Compose(f,gx,y,z)** 返回函数 $f(g(z))$，指定 f 的自变量是 x，指定 g 的自变量为 y。

【例 3-102】 使用 compose 函数求复合方程的解，其运算过程如下。

```
>> syms x y z u v;
>> f=1/(1-x^2);
>> g=sin(sqrt(x));
>> m=tan(x+1);
```

```
>> n=3*x+2 ;
>> compose(f,g);
>> compose(f,g,u);
>> compose(m,n,u,z);
>> compose(m,n,x,y,z);
>> compose(m,n,u,v,z)
```

输出结果如下：

```
ans =
     1/(1-sin(x^(1/2))^2)
ans =
     1/(1-sin(u^(1/2))^2)
ans =
     tan(3*z+3)
ans =
     tan(1+x)
ans =
     tan(3*x+3)
ans =
     tan(1+x)
```

3.7.4 反方程求解

在了解了代数方程、微分方程及复合方程的求解方式以后，有人会产生疑问，那么反方程是否有便捷的求解方式呢？

MATLAB 同样提供 finverse 函数指令来求解反方程，其调用方式如下。

- ❑ **g=finverse(f)** 在 f 函数的反函数存在的情况下，返回 f 函数的反函数自变量为默认变量。
- ❑ **g=finverse(f,v)** 在 f 函数的反函数存在的情况下，返回 f 函数的反函数，自变量设置为 v。

【例 3-103】 计算由反函数构成方程的解，输入指令如下。

```
>> syms x y u v X Y U V;
>> f1=x^3-2*y;
>> f2=exp(2*u+v);
>> g1=finverse(f1,y);
>> g2=finverse(f2,u);
>> g3=finverse(f2,v);
>> X= solve(g1,x);
>> Y= solve(g1,y);
>> U= solve(g2,u);
```

输出结果如下。

```
g1 =
    x^3/2-y/2
g1 =
```

```
      log(u)/2 - v/2
g3 =
      log(v) - v/2
X =
    y^(1/3) -1/2*y^(1/3)+1/2*i*3^(1/2) *y^(1/3) -1/2*y^(1/3)+1/2*i*3^(1/2)
   *y^(1/3)
Y =
    x^3
U =
    1/exp(v)
```

3.8 本章小结

本章首先学习了 MATLAB 的符号计算，它是对未赋值的符号对象（可以是常数、变量、表达式）进行运算和处理。数值型计算会在计算过程中产生舍入误差，而符号计算则可以避免这个问题，符号计算在计算过程中不会出现数值型计算，不存在舍入误差问题。然后学习了符号表达式、运算符号运算精度和符号矩阵的计算。最后学习了符号函数的图形绘制和符号方程的求解。通过本章的学习读者可初步掌握符号计算的方式和使用方法。符号运算与数值计算相同，都是科学研究中的重要内容。运用符号运算可以轻松解决许多公式和关系式的推导问题。

3.9 习题

（1）符号运算与数值运算的区别？

（2）求矩阵 $A=\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$ 的行列式值、非共轭转置和特征值。

（3）符号表达式 $f=2x^2+3x+4$ 与 $g=5x+6$ 的代数运算。

（4）对表达式 $2\sqrt{5}+\pi$ 进行任意精度控制的比较。

（5）求微分方程 $x\dfrac{d^2y}{dx^2}-3\dfrac{dy}{dx}=x^2$，$y(1)=0$，$y(0)=0$ 的解。

第 4 章　MATLAB 图形图像功能

MATLAB 提出了句柄图形学（Handle Graphics）的概念，同时为面向对象的图形处理提供了丰富的工具软件支持。MATLAB 在图形绘制时，其中每个图形元素（如其坐标轴或图形上的曲线、文字等）都是一个独立的对象，用户可以对其中任何一个图形元素进行单独修改，而不影响图形的其他部分，具有这样特点的图形称为矢量化（向量化）的绘图，这种矢量化的绘图要求给每个图形元素分配一个句柄（handle），以后对该图形元素做进一步操作时，只需对该句柄进行操作即可。

MATLAB 进一步定义了三维绘图函数，特别是三维图形显示与照相机参数设置等内容。数据可视化是 MATLAB 一项重要功能，它所提供的丰富绘图功能，使得从繁琐的绘图细节中脱离出来，而能够专心于最关心的本质。通过数据可视化的方法，工程科研人员可以对自己的样本数据的分布、趋势特性有一个直观了解。

本章着重介绍二维图形的画法，读者不仅能掌握二维绘图的基本流程，而且能熟练使用 MATLAB 中相应的绘图命令、函数来绘制二维图形。对三维图形只作简单叙述。其余有关句柄图形学的问题，如窗口特性设置、图形界面设计等项内容将在第 7 章讲述图形界面设计内容时详细介绍。

4.1　二维基本绘图函数

二维图形的绘制是 MATLAB 图形处理的基础。MATLAB 提供了丰富的绘图函数，既可以绘制基本的二维图形，又可以绘制特殊的二维图形。

绘制二维图形的基本步骤如下。

（1）数据准备。准备好绘图需要的横坐标变量和纵坐标变量数据。

（2）设置当前绘图区。在指定的位置创建新的当前绘图区。

（3）绘图。创建坐标轴，指定叠加绘图模式，绘制函数曲线。

（4）设置图形中曲线和标记点。设置线宽、线型、颜色等。

（5）设置坐标轴和网格线属性。将坐标轴的范围设置在指定曲线。

（6）标注图形。在图形中添加标题、坐标轴标注和文字标注等。

（7）保存和导出图形。按指定文件格式、属性保存或导出图形。

4.1.1　line 函数

MATLAB 允许用户在图形窗口的任意位置用绘图命令 line 画直线或折线。

line 函数的常用语法格式如下。

```
line(X,Y)
```

```
line(X,Y,Z)
line(X,Y,Z,'PropertyName',PropertyValue,...)
line('PropertyName',PropertyValue,...) low-level-PN/PV pairs only
h = line(...)
```

其中 X，Y 都是一维数组，line(X,Y)能够把(X(i),Y(i))代表的各点用线段顺次连接起来，从而绘制出一条折线。

line 函数的描述如下。

line 是在现有轴上创建一个直线对象，用户可以定义颜色、宽度、直线类型、标记类型及其他的一些特征。直线函数有两种形式：（1）当使用非正式语法定义矩阵坐标数据时，自动颜色和直线类型循环；（2）当仅仅调用带有属性名/属性值的直线函数。

【例 4-1】 画线函数 line 使用实例。利用函数 line 绘制 y=sinx 的图形。

解：在命令窗口输入以下命令。

```
>>x=0:0.4*pi:2*pi;
>>y=sin(x);
>>line(x,y)
```

运行以上程序代码后，得到如图 4-1 所示的图形。

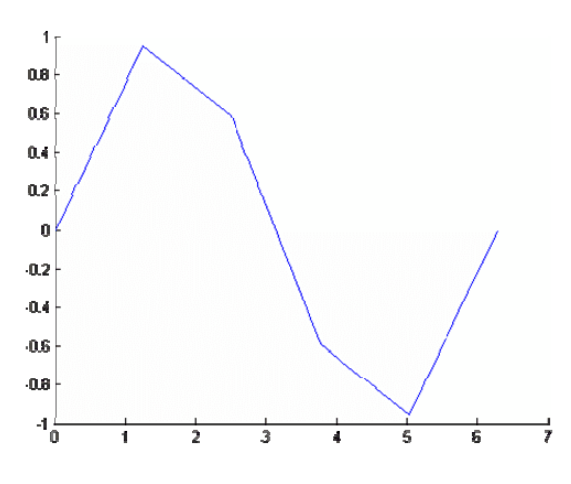


图 4-1　line 函数画线（例 4-1）

4.1.2　semilogx 和 semilogy 函数

在很多工程问题中，通过对数据进行对数转换可以更清晰地看出数据的某些特征，在对数坐标系中描绘数据点的曲线，可以直接地表现对数转换，对数转换有双对数坐标转换和单轴对数坐标转换两种。用 loglog 函数可以实现双对数坐标转换，用 semilogx 和 semilogy 函数可以实现单轴对数坐标转换。

loglog：x 轴和 y 轴均为对数刻度（Logarithmic scale）。

Semilogx：x 轴为对数刻度，y 轴为线性刻度。

Semilogy：x 轴为线性刻度，y 轴为对数刻度。

常用的是 semilogy 函数，即后标为 x 的是在 x 轴取对数，为 y 的是 y 轴坐标取对数。

【例 4-2】 semilogx 函数举例。

解：在命令窗口输入以下命令。

```
>>x = 0:.1:10;
>>y = 2*x+3;
>>semilogy(x,y);
```

运行以上程序代码后，得到如图 4-2 所示的图形。

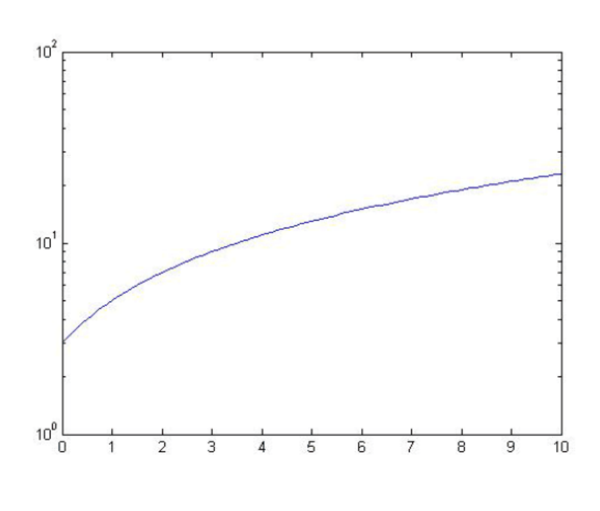


图 4-2 semilogy 举例

4.1.3 logspace 函数

MATLAB 还提供了一个实用的函数：logspace()函数，可按对数等间距地分布来产生一个向量，其调用格式为

```
x=logspace(x1,x2,n)
```

这里，x1 表示向量的起点；x2 表示向量的终点；n 表示需要产生向量点的个数（一般可以不给出，采用默认值 50）。在控制系统分析中一般采用这种方法来构成频率向量 w。关于它的应用后面还要讲到。

4.1.4 plot 函数

plot 函数是 MATLAB 中最核心的二维绘图函数，它有多种语法格式可以实现多种功能。

plot：x 轴和 y 轴均为线性刻度（Linear scale）。

1）最简单的用法 plot(Y)

当 Y 是一维数组时，plot(Y)是把(i,X(i))各点顺次连接起来，其中 i 的取值范围从 1 到 length(X)。

当 Y 是普通的二维数组时，相当于对 Y 的每一列进行 plot(Y(:,i))画线，并把所有的折

线累叠绘制在当前坐标轴下。

2）最常用的用法 plot(X,Y)

plot 最常用的语法格式是接收两个参数的 plot(X,Y)。

当 X 和 Y 都是一维数组时，功能和 line(X,Y)类似，但 plot 函数中的 X 和 Y 也可以是一般的二维数组，这时就是对 X 和 Y 的对应列画线。

特别地，当 X 是一个向量，Y 是一个在某一方向和 X 具有相同长度的二维数组时，plot(X,Y)则是对 X 和 Y 的每一行（或列）画线。

3）拓展的用法 plot(X1,Y1,X2,Y2……Xn,Yn)

对多组变量同时进行绘图了，对于每一组变量，其意义同前所述。

【例 4-3】 plot 函数举例。

解：在命令窗口输入以下命令。

```
>>x=linspace(0, 2*pi, 100);     %100 个点的 x 坐标
>>y=sin(x);                     %对应的 y 坐标
>>plot(x,y);
```

运行以上程序代码后，得到如图 4-3 所示的图形。

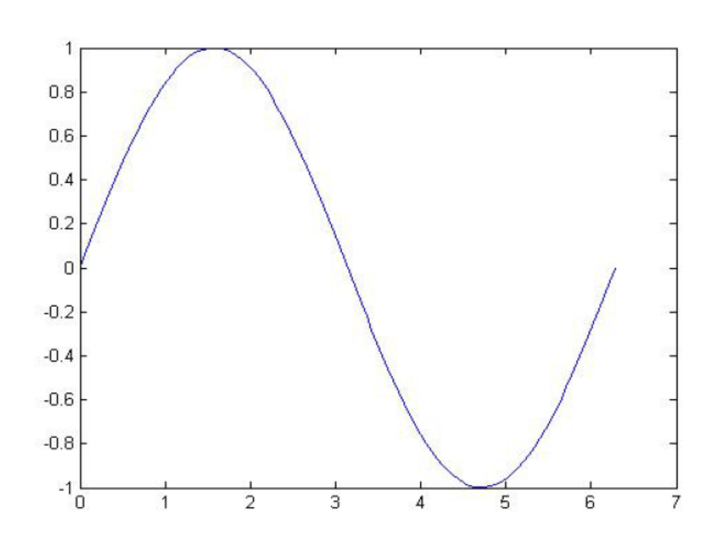


图 4-3　正弦曲线



若要画出多条曲线，只需将坐标对依次放入 plot 函数即可，如

```
plot(x, sin(x), x, cos(x));
```

若要改变颜色，在坐标对后面加上相关字串即可，如

```
plot(x, sin(x), 'c', x, cos(x), 'g');
```

若要同时改变颜色及图线型态（Line style），也是在坐标对后面加上相关字串即可。

plot 是绘制一维曲线的基本函数，但在使用此函数之前，需先定义曲线上每一点的 x 及 y 坐标，下例可画出一条正弦曲线。



【例 4-4】 绘制正弦曲线。

解：在命令窗口输入以下命令。

```
>>x=linspace(0, 2*pi, 100);     %100 个点的 x 坐标
>>y=sin(x);                     %对应的 y 坐标
>>plot(x, sin(x), x, cos(x));
```

运行以上程序代码后，得到如图 4-4 所示的图形。

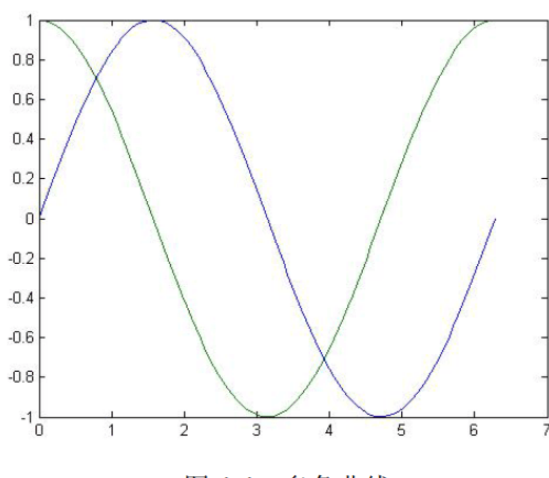


图 4-4　多条曲线

若要改变颜色，在坐标对后面加上相关字串即可，相关颜色符号含义如表 4-1 所示。

表 4-1　颜色符号相关表

颜色符号	含　义	数据点型	含　义	线　型	含　义
b	蓝色	.	点	-	实线
g	绿色	x	X 符号	:	点线
r	红色	+	+号	-.	点划线
c	篮绿色	h	六角星形	--	虚线
m	紫红色	*	星号	(空白)	不画线
y	黄色	s	方形		
k	黑色	d	菱形		



注意

（1）表示属性的符号必须放在同一个字符串中；

（2）可同时指定 2~3 个属性；

（3）与先后顺序无关；

（4）指定的属性中，同一种属性不能有两个以上。

【例 4-5】 绘制改变颜色的曲线。

解：在命令窗口输入以下命令。

```
>>x=linspace(0, 2*pi, 100);     %100 个点的 x 坐标
>>y=sin(x);                     %对应的 y 坐标
>>plot(x, sin(x), 'c', x, cos(x), 'g');
```

运行以上程序代码后，得到如图 4-5 所示的图形。

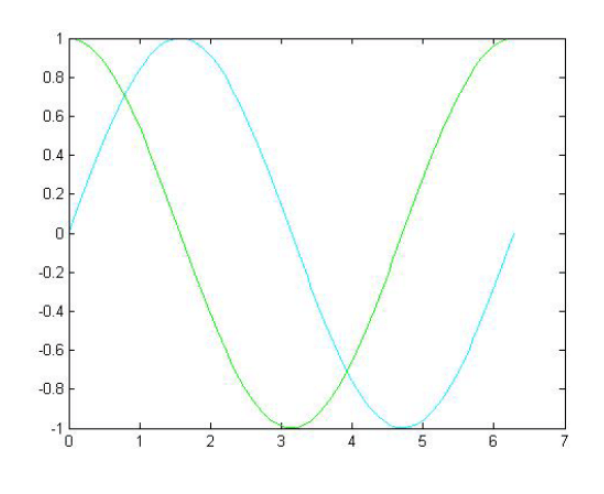


图 4-5　改变颜色后的线条

若要同时改变颜色及图线型态（Line style），也是在坐标对后面加上相关字串即可。

【例 4-6】 改变颜色和线条形态。

解：在命令窗口输入以下命令。

```
>>x=linspace(0, 2*pi, 100);     %100 个点的 x 坐标
>>y=sin(x);                     %对应的 y 坐标
>>plot(x, sin(x), 'co', x, cos(x), 'g*');
```

运行以上程序代码后，得到如图 4-6 所示的图形。

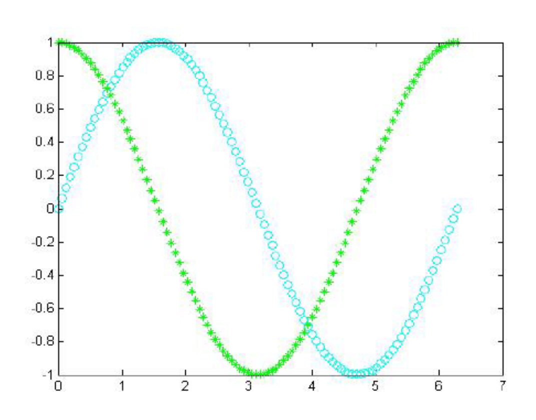


图 4-6　同时改变颜色和线条形态的曲线

4.1.5　plotyy 函数

plotyy 用来绘制双纵坐标图，调用格式如下。

- **plotyy(X1,Y1,X2,Y2)**　以左、右不同纵轴绘制 X1-Y1、X2-Y2 两条曲线。
- **plotyy(X1,Y1,X2,Y2,FUN1)**　以左、右不同纵轴把 X1-Y1、X2-Y2 两条曲线绘制成。
- **plotyy(X1,Y1,X2,Y2,FUN1,FUN2)**　以左、右不同纵轴把 X1-Y1、X2-Y2 两条曲线绘制成 FUN1、FUN2 指定的不同形式的两条曲线。
- **[AX,H1,H2]=plotyy(...)**　返回 AX 中创建的两个坐标轴的句柄及 H1 和 H2 中每个图形绘图对象的句柄。AX(1)为左侧轴，AX(2)为右侧轴。



（1）左纵轴用于 X1-Y1 数据对，右纵轴用于 X2-Y2 数据对。

（2）轴的范围、刻度都自动产生。如果要人工设置，必须使用 axis 函数。

（3）FUN、FUN1、FUN2 可以是 MATLAB 中所有接受 X-Y 数据对的二维绘图指令，如 plot、semilogx、loglog 等函数。

【例 4-7】 plotyy 函数曲线。

解：在命令窗口输入以下命令。

```
>>x1=0:pi/100:2*pi;
>>x2=0:pi/100:3*pi;
>>y1=2*exp(-0.5*x1).*sin(2*pi*x1);
>>y2=1.5*exp(-0.1*x2).*sin(x2);
>>plotyy(x1,y1,x2,y2);
```

运行以上程序代码后，得到如图 4-7 所示的图形。

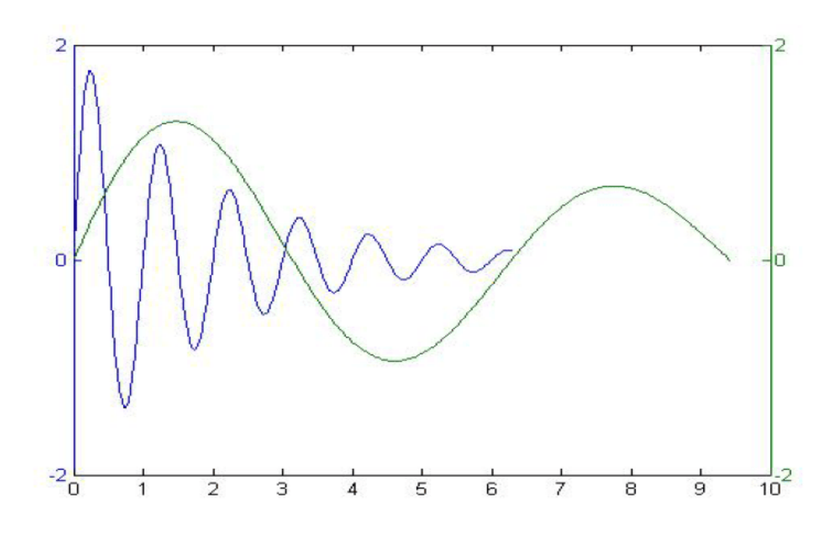


图 4-7　plotyy 函数举例

4.1.6　axis 函数

图形完成后，可用 axis([xmin,xmax,ymin,ymax])函数来调整图轴的范围。

控制坐标性质的 axis 函数的多种调用格式如下。

- **axis([xmin,xmax,ymin,ymax])**　指定二维图形 x 和 y 轴的刻度范围。
- **axis auto**　设置坐标轴的自动刻度(缺省值)。
- **axis manual(或 axis(asix))**　保持刻度不随数据的大小而变化。
- **axis tight**　以数据的大小为坐标轴的范围。
- **axis ii**　设置坐标轴的原点在左上角，i 为纵坐标，i 为横坐标。
- **axis xy**　设置坐标轴回到直角坐标系。
- **axis equal**　设置坐标轴刻度增量相同。
- **axis square**　设置坐标轴长度相同，但刻度增量未必相同。

- **axis normal**　自动调节轴与数据的外表比例，使其他设置生效。
- **axis off**　使坐标轴消隐。
- **axis on**　显现坐标轴。

【例 4-8】 绘制 axis 函数曲线。

解：在命令窗口输入以下命令。

```
>>x=linspace(0, 2*pi, 100); %100 个点的 x 坐标
>>y=sin(x); %对应的 y 坐标
>>plot(x, sin(x), 'co', x, cos(x), 'g*');
>>axis([0, 6, -1.2, 1.2]);
```

运行以上程序代码后，得到如图 4-8 所示的图形。

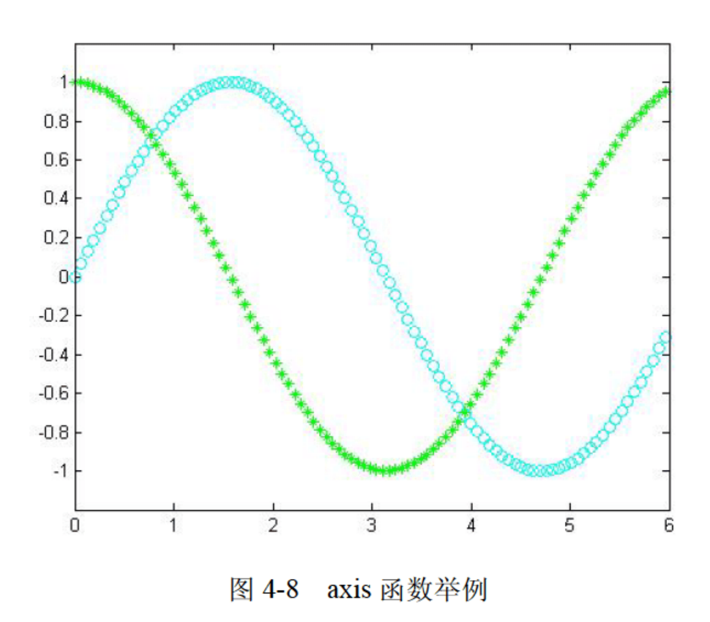


图 4-8　axis 函数举例

当在一个坐标系上画多幅图形时，为区分各个图形，MATLAB 提供了图例的注释说明函数，其格式为

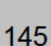
```
legend(字符串 1,字符串 2,字符串 3，…,参数)
```

【例 4-9】 绘制带注释的图形。

解：在命令窗口输入以下命令。

```
>>x=linspace(0, 2*pi, 100);                  %100 个点的 x 坐标
>>y=sin(x);                                  %对应的 y 坐标
>>plot(x, sin(x), 'co', x, cos(x), 'g*');
>>xlabel('Input Value');                     %x 轴注解
>>ylabel('Function Value');                  %y 轴注解
>>title('Two Trigonometric Functions');      %图形标题
>>legend('y = sin(x)','y = cos(x)');         %图形注解
>>grid on;                                   %显示格线
```

运行以上程序代码后，得到如图 4-9 所示的图形。



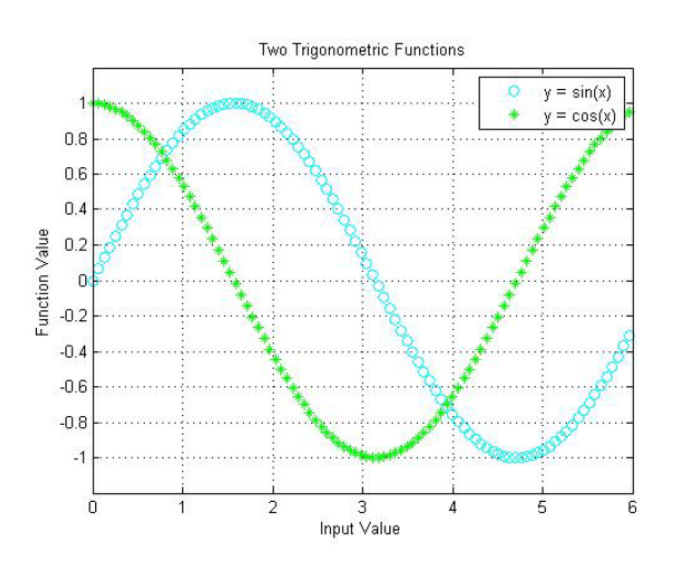


图 4-9 带注释的图形

4.1.7 subplot 函数

在一个图形窗口中绘制多幅图的另一种方法是利用子图绘制函数 subplot()将当前窗口分割成几个区域，然后在各区域中分别绘图。subplot()函数的使用方法如下。

subplot 最常用的语法格式为

```
subplot(m,n,i)
```

表示在当前绘图区中建立 m 行 n 列绘图子区，并在编号为 i 的位置上建立坐标系，并设置该位置为当前绘图区。绘图区的编号优先从顶行开始，然后是第二行，第三行……

【例 4-10】 绘制 subplot 函数曲线。

解：在命令窗口输入以下命令。

```
>>subplot(2,2,1); plot(x, sin(x));
>>subplot(2,2,2); plot(x, cos(x));
>>subplot(2,2,3); plot(x, sinh(x));
>>subplot(2,2,4); plot(x, cosh(x));
```

运行以上程序代码后，得到如图 4-10 所示的图形。

4.1.8 其他特殊函数

MATLAB 还有其他各种二维绘图函数，以适合不同的应用，如表 4-2 所示。

matlab 中函数 bar(x)可以绘制直方图，这对统计或数据采集来说直观实用；bar(x,y)中的 x 必须单调递增或递减，y 为 n×m 矩阵，可视化结果为 m 组，每组 n 个垂直柱，也就是把 y 的行画在一起，同一列的数据用相同的颜色表示；bar(x,y,width)(或 bar(y,width))指定每个直方条的宽度，如 width>1，则直方条会重叠，默认值为 width=0.8；bar(…,’grouped’)使同一组直方条紧紧靠在一起；bar(…,’stack’) 把同一组数据描述在一个直方条上。

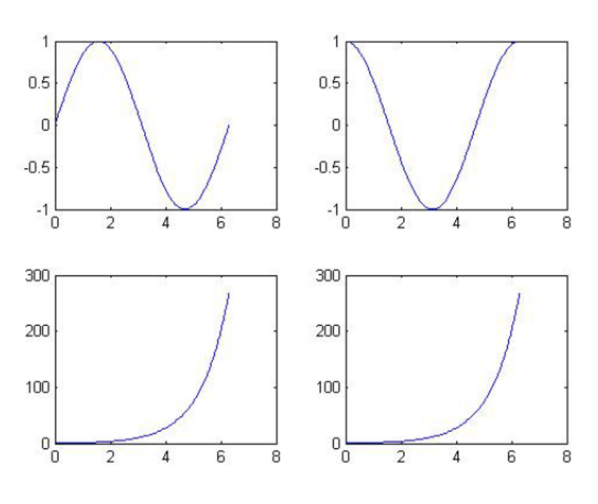


图 4-10　subplot 函数举例

表 4-2　二维绘图函数

1	bar	长条图
2	errorbar	图形加上误差范围
3	fplot	较精确的函数图形
4	polar	极坐标图
5	hist	累计图
6	rose	极坐标累计图
7	stairs	阶梯图
8	stem	针状图
9	fill	实心图
10	feather	羽毛图
11	compass	罗盘图
12	quiver	向量场图

【例 4-11】 绘制直方图。

```
>>y=[5 3 2 9;4 7 2 7;1 5 7 3];
>>plot(2,2),bar(y)
```

其图形如图 4-11 所示。

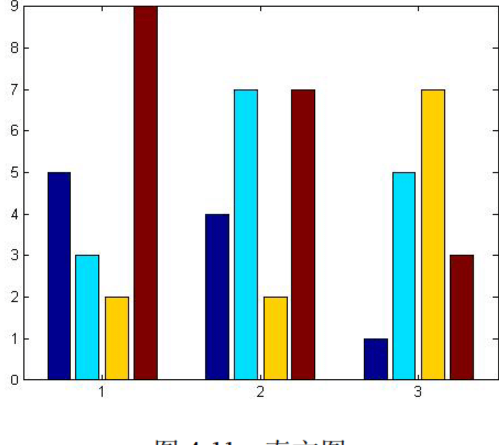


图 4-11　直方图

【例 4-12】 绘制长条图。

```
>>x=1:10;
>>y=rand(size(x));
>>bar(x,y);
```

其图形如图 4-12 所示。

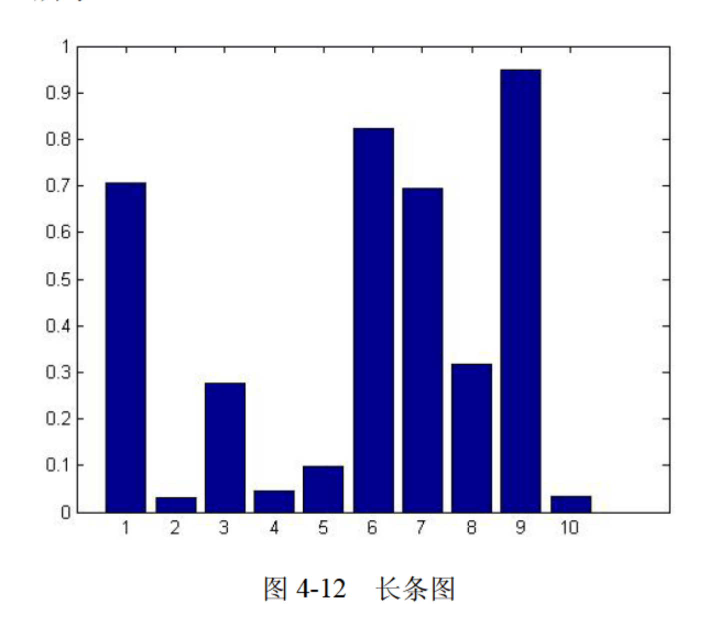


图 4-12　长条图

对于变化剧烈的函数，可用 fplot 进行较精确的绘图，对剧烈变化处进行较密集的取样，fplot(fun,limits)在指定的范围 limits 内画出函数名为 fun 的图像。其中，limits 是一个指定 x 轴范围的向量[xmin xmax]或是 x 轴和 y 轴范围的向量[xmin xmax ymin ymax]。

【例 4-13】 利用 fplot 函数精确绘图。

```
>>fplot('sin(1/x)', [0.02 0.2]);  %[0.02 0.2]是绘图范围
```

其图形如图 4-13 所示。

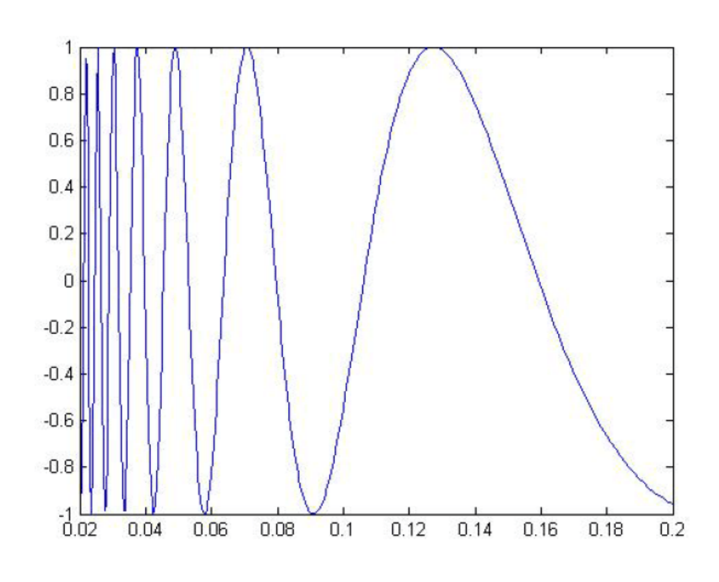


图 4-13　fplot 比较精确的图

若要产生极坐标图形，可用 polar 描绘极坐标图像。最简单且常用的命令格式为 polar(THETA, RHO)，其中，THETA 是用弧度制表示的角度，RHO 是对应的半径。

【例 4-14】 绘制 polar 函数图形。

```
>>theta=linspace(0, 2*pi);
>>r=cos(4*theta);
>>polar(theta, r);
```

其图形如图 4-14 所示。

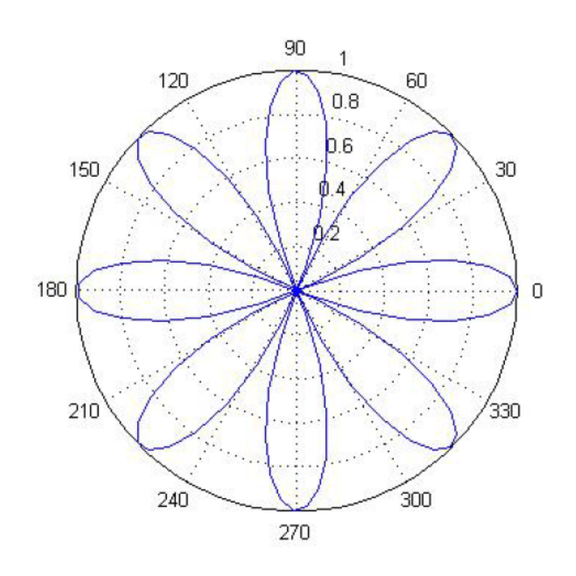


图 4-14　体现 polar 函数的图形

在 MATLAB 中 stairs 函数用于绘制阶梯状图，在图像处理中的直方图均衡化技术中有很大的意义。在 MATLAB 的命令窗口中输入 doc stairs 或 help stairs 即可获得该函数的帮助信息。

调用格式如下。

stairs(Y)

stairs(X,Y)

stairs(...,LineSpec)

stairs(...,'PropertyName',propertyvalue)

stairs(axes_handle,...)

h = stairs(...)

[xb,yb] = stairs(Y,...)

【例 4-15】 绘制梯形图。

```
>>x=linspace(0,10,50);
>>y=sin(x).*exp(-x/3);
>>stairs(x,y);
```

其图形如图 4-15 所示。

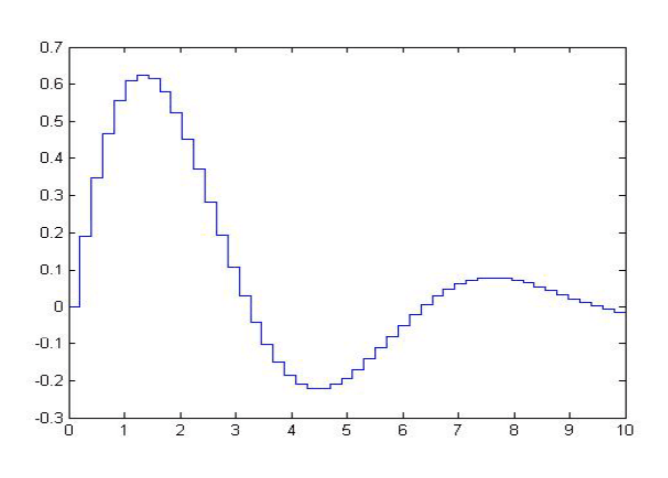


图 4-15　梯形图

fill 将数据点视为多边形顶点，并将此多边形涂上颜色。

fill 函数具体调用方法如下。

fill(X,Y,C)，创建填充多边形顶点的颜色，从 C 中指定的 X 和 Y 的数据是一个向量或矩阵，用于作为颜色表索引。如果 C 是一个行向量，则要求 C 的维数等于 X 和 Y 的列数；如果 C 是一个列向量，则要求 C 的维数与 X 和 Y 的维数相等。如果有必要，填充闭合多边形的顶点连接。

fill(X,Y,ColorSpec)，二维多边形填充指定的 X 和 Y 与指定的 ColorSpec 颜色。

fill(X1,Y1,C1,X2,Y2,C2,...)，指定多个二维填充区域。

fill(...,'PropertyName',PropertyValue)，允许用户指定一个补丁的图形对象的属性名称和值。

h = fill(...)，返回一个向量处理图形对象的补丁，每个补丁图形对象对应一个柄。

【例 4-16】 绘制涂色图。

```
>>x=linspace(0,10,50);
>>y=sin(x).*exp(-x/3);
>>fill(x,y,'b'); %'b'为蓝色
```

其图形如图 4-16 所示。

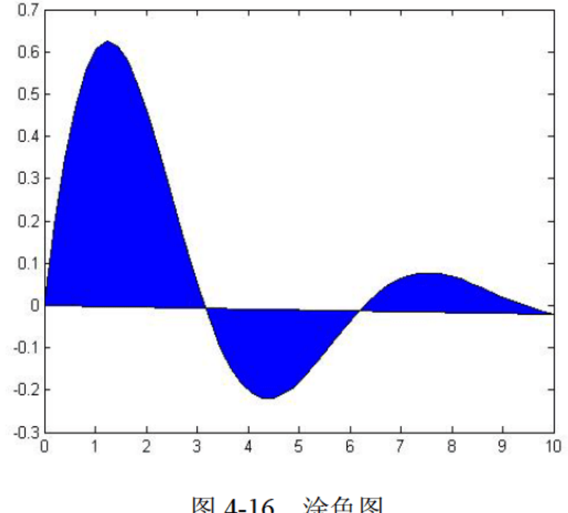


图 4-16　涂色图

feather 复平面图形，把复数矩阵中元素的相角和幅值显示成沿横轴等间隔辐射的箭头，格式 feather(z)、feather(x,y)等价于 feather(x+y*i)、feather(z,str)、str 是确定的线形绘制箭头。feather 将每一个数据点视作复数，并以箭号画出。

【例 4-17】 feather 函数绘图 1。

```
>>x=1:0.01:20;
>>feather(x)
```

其图形如图 4-17 所示。

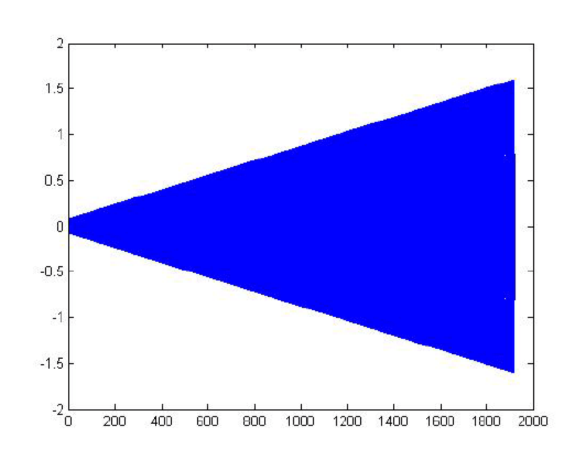


图 4-17 feather 函数举例（1）

【例 4-18】 feather 函数绘图 2。

```
>>theta=linspace(0, 2*pi, 20);
>>z = cos(theta)+i*sin(theta);
>>feather(z);
```

其图形如图 4-18 所示。

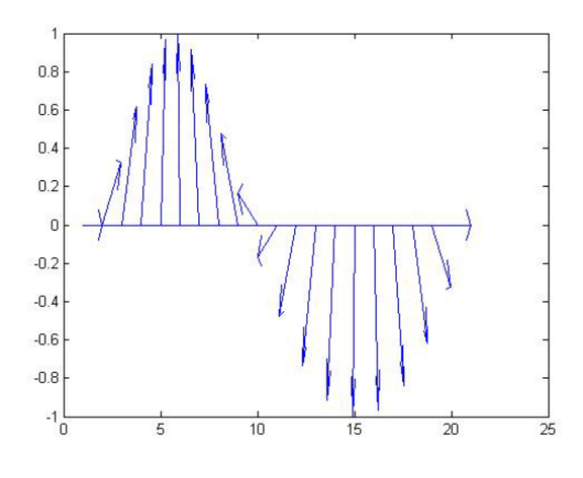


图 4-18 feather 函数举例（2）

compass 函数用法如下。

（1）compass(x,y)：函数绘制一个由原点出发、由(x,y)组成的向量箭头图形。



（2）compass(z)：等价于 compass(real(z),imag(z))。

（3）compass(...,LineSpec)：用参量 LineSpec 指定箭头的线型、标记符号、颜色等属性。

（4）h = compass(...)：函数返回 line 对象的句柄给 h。

【例 4-19】 绘制 compass 函数图形。

```
>>theta=linspace(0, 2*pi, 20);
>>z = cos(theta)+i*sin(theta);
>>compass(z);
```

其图形如图 4-19 所示。

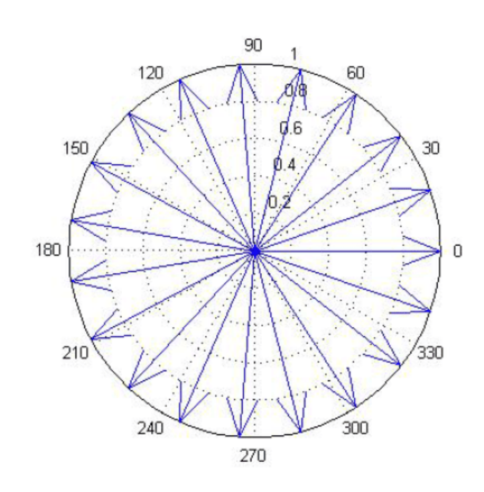


图 4-19　体现 compass 函数的图形

4.2　三维基本图形

MATLAB 具有强大的三维绘图能力，如绘制三维曲线、三维网格图和三维曲面图，并提供了大量的三维绘图函数。

绘制三维图形的基本步骤如下。

（1）准备数据。

（2）设置当前绘图区。

（3）调用绘图指令。

（4）设置视角。

（5）设置图形的曲线和标记点的形式。

（6）保存并导出图形。

三维绘图指令如表 4-3 所示。

表 4-3　创建线程其他系统函数

类　别	指　令	说　明
网状图	mesh, ezmesh	绘制立体网状图
	meshc, ezmeshc	绘制带有等高线的网状图
	meshz	绘制带有“围裙”的网状图

续表

类　别	指　令	说　明
曲面图	surf, ezsurf	立体曲面图
	surfc, ezsurfc	绘制带有等高线的曲面图
	surfl	绘制带有光源的曲面图
曲线图	plot3, ezplot3	绘制立体曲线图
底层函数	surface	Surf 函数用到的底层指令
	line3	plot3 函数用到的底层指令
等高线	contour3	绘制等高线
水流效果	waterfall	在 x 方向或 y 方向产生水流效果
影像表示	pcolor	在二维平面中以颜色表示曲面的高度

三维绘图的主要功能如下。

- ❑ 绘制三维线图。
- ❑ 绘制等高线图。
- ❑ 绘制伪彩色图。
- ❑ 绘制三维网线图。
- ❑ 绘制三维曲面图、柱面图和球面图。
- ❑ 绘制三维多面体并填充颜色。

基本 XYZ 立体绘图命令如下。

三维曲线与一组(x,y,z)坐标相对应的点连接而成。

绘图格式为

```
plot3(X,Y,Z,'s')
plot3(X1,Y1,Z1,'s1',X2,Y2,Z2,'s2',...)
```

（1）X、Y、Z 是同维向量时，则绘制以 X、Y、Z 元素为 x、y、z 坐标的三维曲线。

（2）X、Y、Z 是同维矩阵时，则以 X、Y、Z 对应列元素为 x、y、z 坐标绘制多条曲线，曲线条数等于矩阵的列数。

（3）(X1,Y1,Z1,'s1')与(X2,Y2,Z2,'s2')的结构与作用和(X,Y, Z,'s')相同，表示同一指令绘两组以上曲线。

（4）s、s1、s2 的意义与二维相同。

【例 4-20】 绘制三维曲线 1。

```
>> x=0: pi/50: 10*pi;
>> y=sin ( x );
>> z=cos ( x );
>> plot3 ( x, y, z )
```

其图形如图 4-20 所示。

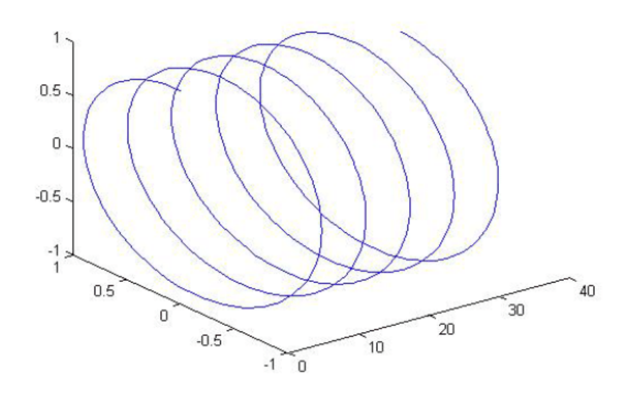


图 4-20　三维曲线（1）

【例 4-21】 绘制三维曲线 2。

```
>>x=linspace(0,pi+pi/6,30) %把 x 分 30 个点，就是数据点
>>y=[1 2 3 4 5 6 7] %角度，假设为 7 个
>>temp=zeros(1,length(x))
>>z=sin(x/2) %幅度
>>for i=1:length(y)
>>y1=y(i)+temp %把角度的一个值，变为 30 个相同的角度值
>>plot3(x,y1,z)
>>grid on
>>hold on
```

其图形如图 4-21 所示。

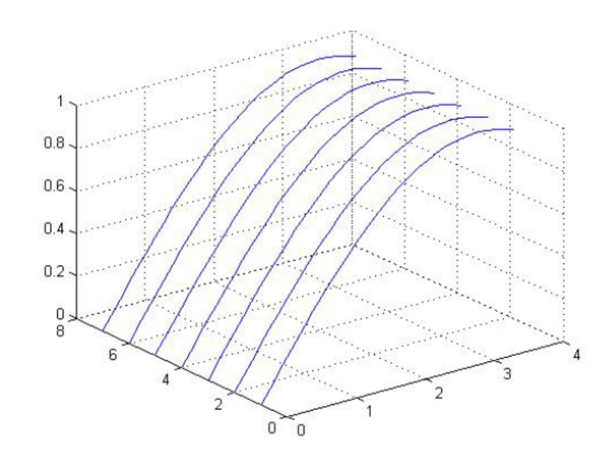


图 4-21　三维曲线（2）

4.2.1　mesh 函数

mesh 函数生成由 X、Y、Z 指定的网线面，由 C 指定颜色的三维网格图。

用法：mesh(X,Y,Z)

（1）若 X 与 Y 均为向量，length(X)=n，length(Y)=m，而[m,n]=size(Z)，空间中的点(X(j),Y(I),Z(I,j)) 为所画曲面网线的交点，X 对应于 Z 的列，Y 对应于 Z 的行。

（2）若 X 与 Y 均为矩阵，则空间中的点 (X(I,j),Y(I,j),Z(I,j))为所画曲面网线的交点。

mesh(Z)：由[n,m] = size(Z)得，X =1:n 与 Y=1:m，其中，Z 为定义在矩形划分区域上的单值函数。

mesh(…,C)：用由矩阵 C 指定的颜色画网线网格图。MATLAB 对矩阵 C 中的数据进行线性处理，以便从当前色图中获得有用的颜色。

mesh(…,PropertyName,PropertyValue, …)：对指定的属性 PropertyName 设置属性值 PropertyValue，可以在同一语句中对多个属性进行设置。

h = mesh(…)：返回 surface 图形对象句柄。

运算规则如下。

（1）数据 X、Y 和 Z 的范围，或者是对当前轴的 XLimMode、YLimMode 和 ZLimMode 属性的设置决定坐标轴的范围。命令 axis 可对这些属性进行设置。

（2）参量 c 的范围，或者是对当前轴的 Clim 和 ClimMode 属性的设置（可用命令 caxis 进行设置），决定颜色的刻度化程度。刻度化颜色值作为引用当前色图的下标。

（3）网格图显示命令生成，由于把 Z 的数据值是用当前色图表现出来的颜色值。MATLAB 会自动用最大值与最小值计算颜色的范围（可用命令 caxis auto 进行设置），最小值用色图中的第一个颜色表现，最大值用色图中的最后一个颜色表现。MATLAB 会对数据的中间值执行一个线性变换，使数据能在当前的范围内显示出来。

【例 4-22】 单位矩阵的网图。

```
>> a = eye ( 20 );
>> mesh ( a )
```

其图形如图 4-22 所示。

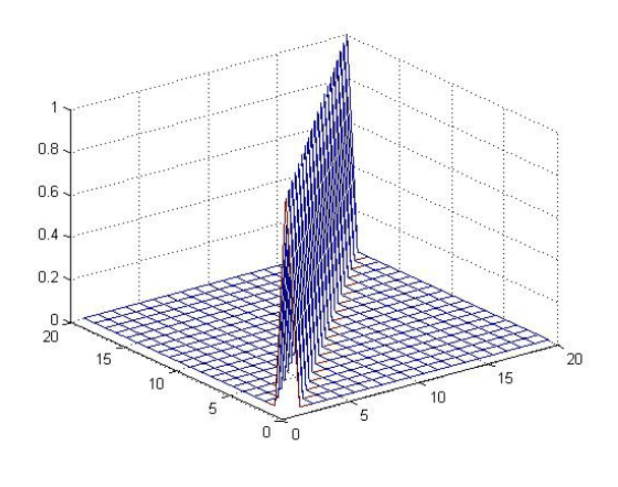


图 4-22 mesh 函数举例

【例 4-23】 画出由函数 $z = xe^{-(x^2+y^2)}$ 形成的立体网状图。

```
>>x=linspace(-2, 2, 25);         %在 x 轴上取 25 点
>>y=linspace(-2, 2, 25);         %在 y 轴上取 25 点
>>[xx,yy]=meshgrid(x, y);        %xx 和 yy 都是 25x25 的矩阵
>>zz=xx.*exp(-xx.^2-yy.^2);      %计算函数值，zz 也是 21x21 的矩阵
```

```
>>mesh(xx, yy, zz); %画出立体网状图
```

其图形如图 4-23 所示。

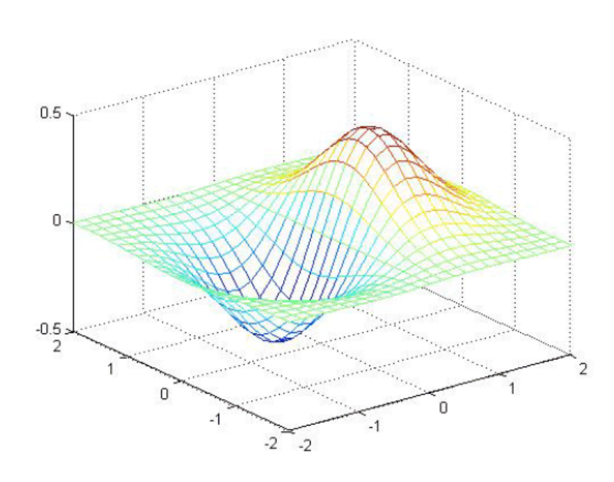


图 4-23　立体网状图

4.2.2　surf 函数

surf 和 mesh 的用法类似，surf 和 surfc 是通过矩形区域来观测数学函数的函数。surf 和 surfc 能够产生由 X、Y、Z 指定的有色参数化曲面，即三维有色图。

用法：surf(Z) 生成一个由矩阵 Z 确定的三维带阴影的曲面图，其中 [m,n] = size(Z)，而 X = n，Y = 1:m。高度 Z 为定义在一个几何矩形区域内的单值函数，Z 同时指定曲面高度数据的颜色，所以颜色对于曲面高度是恰当的。

surf(X,Y,Z) 数据 Z 同时为曲面高度，也是颜色数据。X 和 Y 为定义 X 坐标轴和 Y 坐标轴的曲面数据。若 X 与 Y 均为向量，length(X)=n，length(Y)=m，而[m,n]=size(Z)，在这种情况下，空间曲面上的节点为(X(I),Y(j),Z(I,j))。

surf(X,Y,Z,C) 用指定的颜色 C 画出三维网格图。MATLAB 会自动对矩阵 c 中的数据进行线性变换，以获得当前色图中可用的颜色。

surf(…,PropertyName',PropertyValue) 对指定的属性 PropertyName 设置为属性值 PropertyValue。

h = surf(…) 返回一个 surface 图形对象句柄给变量 h。

运算规则如下。

（1）严格地讲，一个参数曲面是由两个独立的变量 I、j 来定义的，它们在一个矩形区域上连续变化。例如，a<=I<=b,c<=j<=d，三个变量 X、Y、Z 确定了曲面，曲面颜色由第 4 参数矩阵 C 确定。

（2）矩形定义域上的点有如下关系。

```
              A(I-1,j)
                 |
B(I,j-1) ---- C(I,j) ---- D(I,j+1)
                 |
              E(I+1,j)
```

这个矩形坐标方格对应于曲面上的有四条边的块，在空间的点的坐标为[X(J),Y(J,Z)]，每个矩形内部的点根据矩形的下标和相邻的四个点连接；曲面上的点只有相邻的三个点，曲面上四个角上的点只有两个相邻点，上面这些定义了一个四边形的网格图。

（3）曲面颜色可以有两种方法来指定：指定每个节点的颜色或者是每一块的中心点颜色。在一般的设置中，曲面不一定为变量 X 和 Y 的单值函数，进一步而言，有四边的曲面块不一定是平面的，可以用极坐标、柱面坐标和球面坐标定义曲面。

（4）命令 shading 设置阴影模式。若模式为 interp，C 必须与 X、Y、Z 同型；它指定了每个节点的颜色，曲面块内的颜色由附近几个点的颜色用双线性函数计算出来。若模式为 facted(缺省模式)或 flat，c(I,j)指定曲面块中的颜色如下。

```
A(I,j)----------- B (I,j+1)
  |      C(I,j)      |
C(I+1,j) --------- D(I+1,j)
```

在这种情形下，C 可以与 X、Y 和 Z 同型，且它的最后一行和最后一列将被忽略，换句话说，就是 C 的行数和列数可以比 X，Y，Z 少 1。

（5）命令 surf 将指定图形视角为 view(3)。

（6）数据 X、Y、Z 的范围或通过对坐标轴的属性 XlimMode、YlimMode 和 ZlimMode 的当前设置（可以通过命令 axis 来设置），将决定坐标轴的标签。

（7）参数 C 的范围或通过对坐标轴的属性 Clim 和 ClimMode 的设置（可以通过命令 caxis 来设置），将决定颜色刻度化。刻度化的颜色值将作为引用当前色图的下标。

【例 4-24】 立体曲面图 1。

```
>> [ X, Y ] = meshgrid ( [ -4: 0.5: 4] ) ;
>> Z = sqrt ( X.^2+Y.^2 );
>> surf ( Z )
```

其图形如图 4-24 所示。

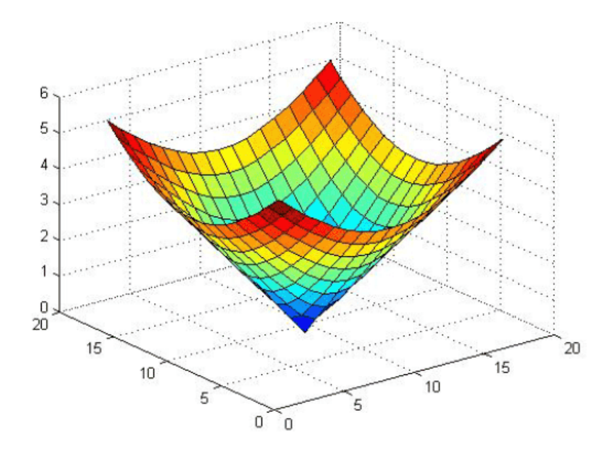


图 4-24　立体曲面图（1）

meshgrid 为形成网格，可以把 X 和 Y 轴分开，如：meshgrid([-1:0.1:1],[-2:0.1:2])。

【例 4-25】 立体曲面图 2。

```
>>x=linspace(-2, 2, 25);    %在 x 轴上取 25 点
>>y=linspace(-2, 2, 25);    %在 y 轴上取 25 点
>>[xx,yy]=meshgrid(x, y);   %xx 和 yy 都是 25x25 的矩阵
```

```
>>zz=xx.*exp(-xx.^2-yy.^2); %计算函数值，zz 也是 25x25 的矩阵
>>surf(xx, yy, zz); %画出立体曲面图
```

其图形如图 4-25 所示。

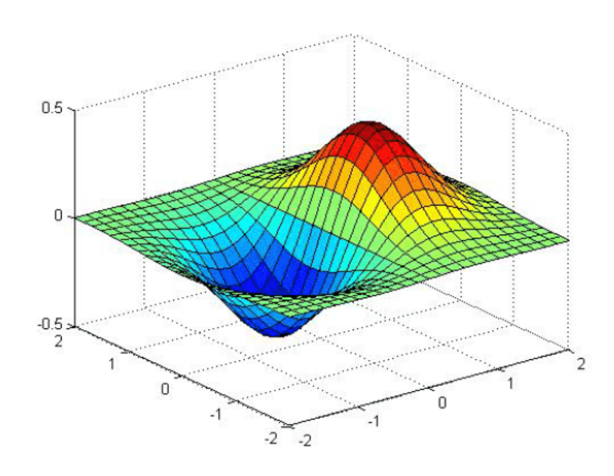


图 4-25　立体曲面图（2）

4.2.3　peaks 函数

为了方便测试立体绘图，MATLAB 提供了一个 peaks 函数，可产生一个凹凸有致的曲面，包含三个局部极大点及三个局部极小点，其方程式为

$$y = 3(1-x)^2 e^{-x^2(y+1)^2} - 10(\frac{x}{5} - x3 - y5)e^{-x^2-y^2} - \frac{1}{3}e^{-(x+1)^2-y^2}$$

要画出此函数的最快方法即是直接键入 peaks。

【例 4-26】 绘制 peaks 函数的图形。

```
>>peaks
>>z = 3*(1-x).^2.*exp(-(x.^2) - (y+1).^2) - 10*(x/5 - x.^3
        -y.^5).*exp(-x.^2-y.^2) - 1/3*exp(-(x+1).^2 - y.^2)
```

其图形如图 4-26 所示。

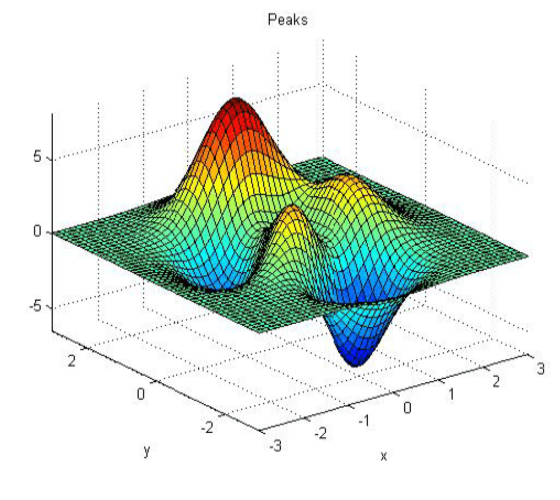


图 4-26　体现 peaks 函数的图形

用户亦可对 peaks 函数取点，再以各种不同方法进行绘图。meshz 可将曲面加上围裙。

【例 4-27】 加围裙的图形。

```
>>[x,y,z]=peaks;
>>meshz(x,y,z);
```

其图形如图 4-27 所示。

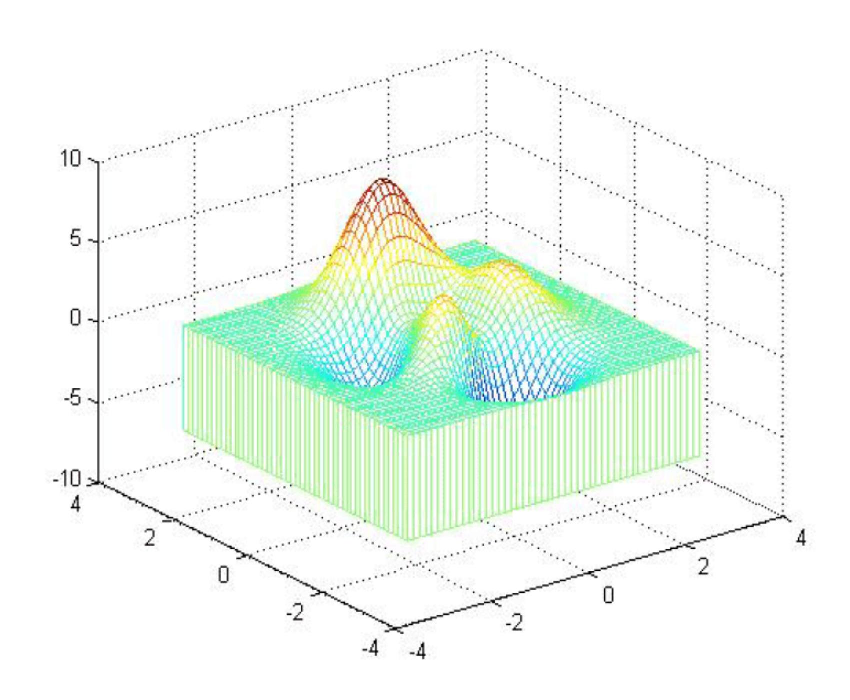


图 4-27 加围裙的图形

waterfall 可在 x 方向或 y 方向产生水流效果。

功能：瀑布图。

用法：waterfall(X,Y,Z) 用参数 X、Y 与 Z 的数据画“瀑布”效果图。若 X 与 Y 都是向量，则 X 与 Z 的列相对应，Y 与 Z 的行相对应，即 length(X)=Z 的列数，length(Y)=Z 的行数。参数 X 与 Y 定义了 x 轴与 y 轴，Z 定义了 z 轴的高度，Z 同时确定了颜色，所以颜色能恰当反映曲面的高度。若想研究数据的列，可以输入 waterfall(Z')或 waterfall(X',Y',Z')。

waterfall(Z) 画出一瀑布图，其中缺省地有 X=1:Z 的行数，Y=1:Z 的行数，且 Z 同时确定颜色，所以颜色能恰当地反映曲面高度。

waterfall(…,C) 用比例化的颜色值从当前色图中获得颜色，参量 C 决定颜色的比例，为此，必须与 Z 同型。系统使用线性变换，从当前色图中获得颜色。

h = waterfall(…) 返回 patch 图形对象的句柄 h，用于画出图形。

【例 4-28】 x 方向流水图。

```
>>[x,y,z]=peaks;
>>waterfall(x,y,z);
```

其图形如图 4-28 所示。

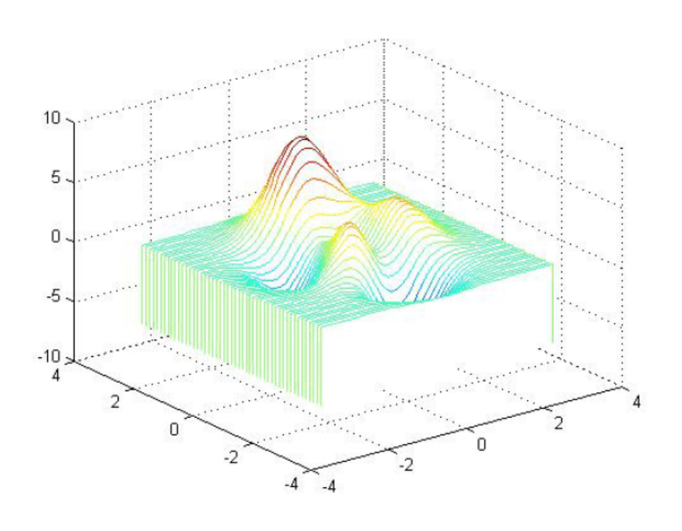


图 4-28　x 方向产生水流效果

下列命令产生在 y 方向的水流效果。

【例 4-29】 y 方向流水图。

```
>>[x,y,z]=peaks;
>>waterfall(x',y',z');
```

其图形如图 4-29 所示。

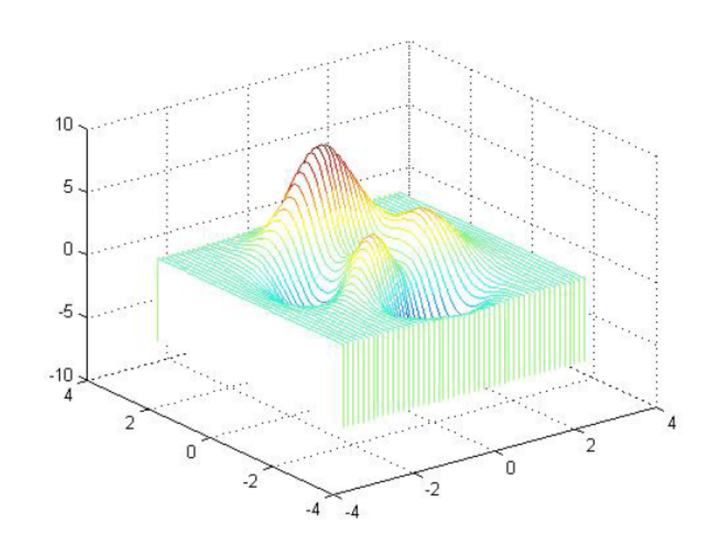


图 4-29　y 方向产生水流效果

meshc 同时画出网状图与等高线。

【例 4-30】 同时绘制网状图和等高线。

```
>>[x,y,z]=peaks;
>>meshc(x,y,z);
```

其图形如图 4-30 所示。

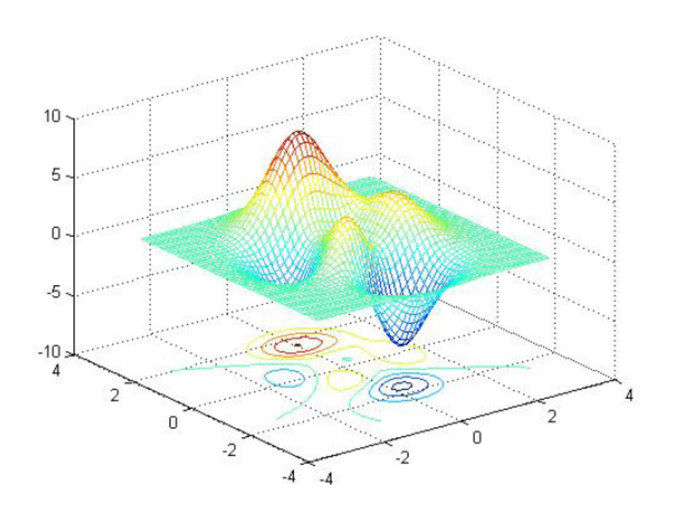


图 4-30　网状图与等高线

Surfc 可同时画出曲面图与等高线。

【例 4-31】 同时绘制曲面图和等高线。

```
>>[x,y,z]=peaks;
>>surfc(x,y,z);
```

其图形如图 4-31 所示。

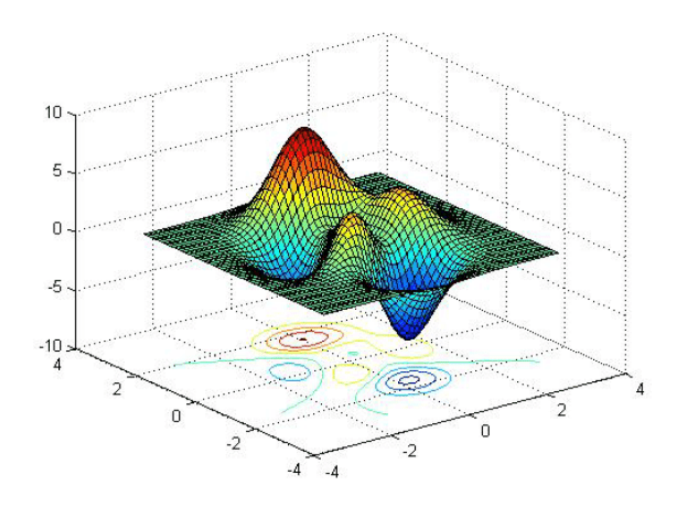


图 4-31　曲面图与等高线

contour3 画出曲面在三度空间中的等高线。

【例 4-32】 三度空间中的等高线。

```
>>contour3(peaks, 20);
```

其图形如图 4-32 所示。

contour 画出曲面等高线在 XY 平面的投影。

contour 命令的常用调用格式如下。

（1）contour(z)变量 z 就是需要绘制的等高线函数表达式。

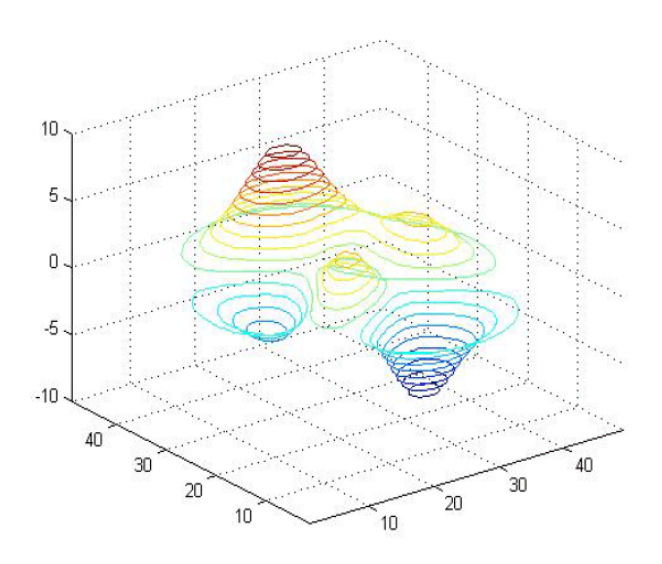


图 4-32　三度空间的等高线

（2）contour(z,n)中参数 n 是所绘图形等高线的条数。

（3）contour(z,v)中参数 v 是一个输入向量，等高线的条数等于该向量的长度，而且等高线的数值等于对应向量的数值元素[c,h]=contour(...)，c 是等高线矩阵，h 是等高线句柄。

【例 4-33】 绘制等高线在 XY 平面的投影。

```
>>contour(peaks, 20);
```

其图形如图 4-33 所示。

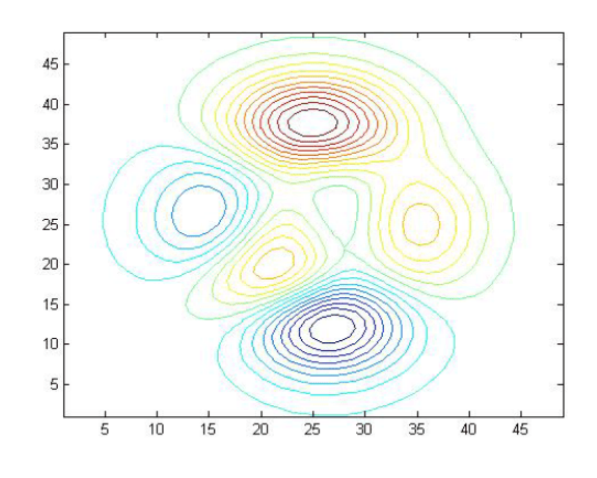


图 4-33　等高线在 XY 平面的投影

plot3 可画出三度空间中的曲线。

【例 4-34】 绘制三度空间中的曲线。

```
>>t=linspace(0,20*pi, 501);
>>plot3(t.*sin(t), t.*cos(t), t);
```

其图形如图 4-34 所示。

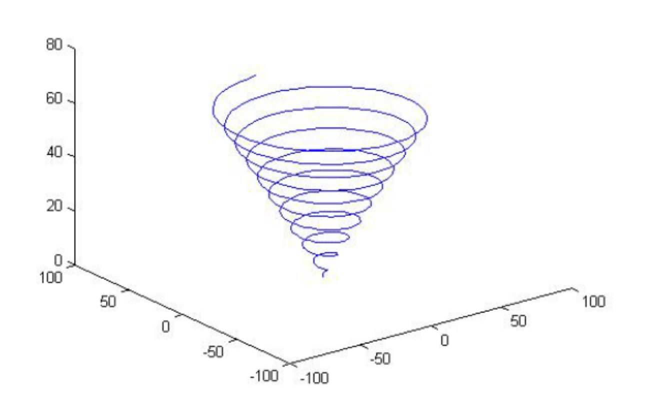


图 4-34　三度空间中的曲线

同时画出两条三度空间中的曲线。

【例 4-35】 绘制两条三度空间中的曲线。

```
>>t=linspace(0, 10*pi, 501);
>>plot3(t.*sin(t), t.*cos(t), t, t.*sin(t), t.*cos(t), -t);
```

其图形如图 4-35 所示。

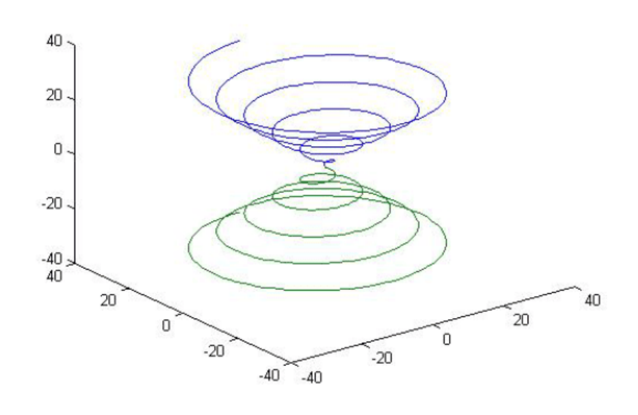


图 4-35　两条三度空间中的曲线

4.2.4　特殊函数

1．饼图：pie3

用于表示矢量或矩阵中各元素所占有的比例。函数 pie 和 pie3 提供平面饼图和三维饼图的绘图功能。

pie(x) 使用 x 中的数据绘制饼图，x 中的每一个元素用饼图中的一个扇区表示。

pie(x,explode) 绘制向量 x 的饼图，如果向量 x 的元素和小于 1，则绘制不完全的饼图。explode 为一个与 x 尺寸相同的矩阵，其非零元素所对应的 x 矩阵中的元素从饼图中分离出来。

三维饼图：有一定厚度的饼图，由函数 pie3 实现，调用方法与二维饼图相同。

【例 4-36】 绘制饼图。

```
>>pie3([2,3,4])  %2/(2+3+4)=0.22，3/(2+3+4)=0.33，4/(2+3+4)=0.44
```

其图形如图 4-36 所示。

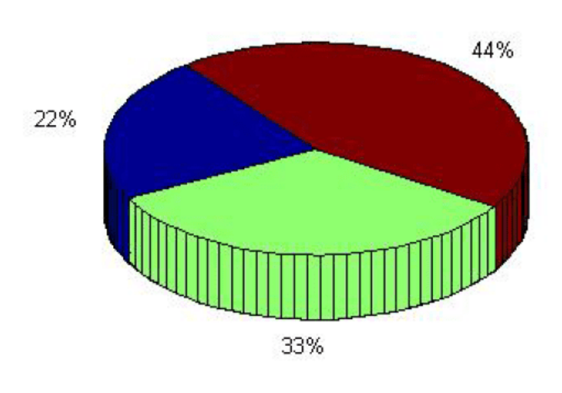


图 4-36　饼图

2．柱面图：cylinder

功能：生成圆柱图形，该命令生成一单位圆柱体的 x、y、z 轴的坐标值。用户可以用命令 surf 或命令 mesh 画出圆柱形对象，或者用没有输出参量的形式立即画出图形。

用法：[X,Y,Z] = cylinder 返回一半径为 1、高度为 1 的圆柱体的 x、y、z 轴的坐标值，圆柱体的圆周有 20 个距离相同的点。

[X,Y,Z] = cylinder(r) 返回一半径为 r、高度为 1 的圆柱体的 x、y、z 轴的坐标值，圆柱体的圆周有 20 个距离相同的点。

[X,Y,Z] = cylinder(r,n) 返回一半径为 r、高度为 1 的圆柱体的 x、y、z 轴的坐标值，圆柱体的圆周有指定的 n 个距离相同的点。

cylinder(…) 没有任何的输出参量，直接画出圆柱体。

【例 4-37】 绘制柱面图。

```
>> cylinder([2,3,4,5])
```

其图形如图 4-37 所示。

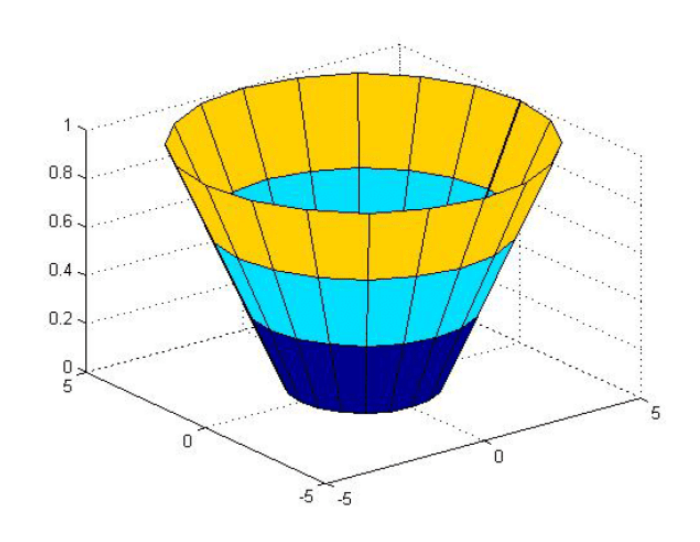


图 4-37　柱面图

3．球面图：sphere

功能：生成球体。

用法：sphere 生成三维直角坐标系中的单位球体，该单位球体由 20×20 个面组成。

sphere(n) 在当前坐标系中画出有 n×n 个面的球体。

[X,Y,Z] = sphere(n) 返回三个阶数为(n+1)×(n+1)的直角坐标系中的坐标矩阵。该命令没有画图，只是返回矩阵。用户可以用命令 surf(X,Y,Z)或 mesh(X,Y,Z)画出球体。

【例 4-38】 绘制球面图。

```
>> sphere(20)
```

其图形如图 4-38 所示。

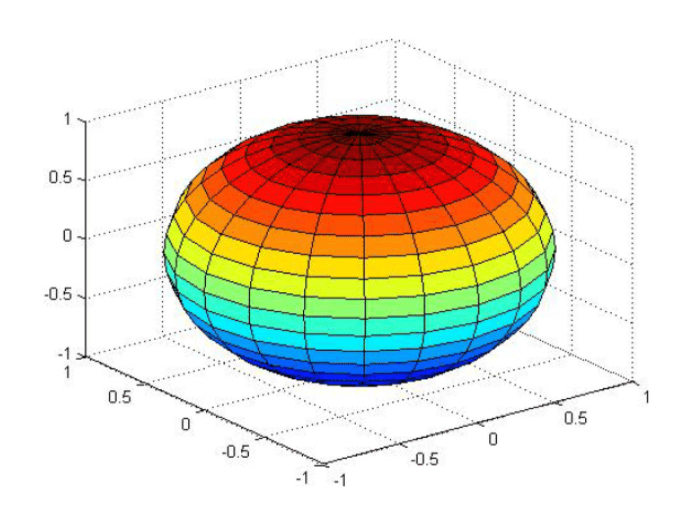


图 4-38　球面图

4.3　图形处理技术

MATLAB 除了提供强大的绘图功能外，还提供了强大的图形处理功能，下面对这些相关的技术进行具体介绍。

4.3.1　坐标轴的调整

1．axis 用法

axis([xmin xmax ymin ymax]) 设置当前坐标轴的 x 轴与 y 轴的范围。

axis([xmin xmax ymin ymax zmin zmax cmin cmax]) 设置当前坐标轴的 x 轴、y 轴与 z 轴的范围，当前颜色刻度范围，该命令也同时设置当前坐标轴的属性，XLim、YLim 与 ZLim 为所给参数列表中的最大值和最小值。另外，坐标轴属性 XLimMode、YLimMode 与 ZLimMode 设置为“manual”。



v = axis 返回一包含 x 轴、y 轴与 z 轴的刻度因子的行向量，其中 v 为四维或六维向量，这取决于当前坐标是二维还是三维的。返回的值包含当前坐标轴的 XLim、YLim 与 ZLim 属性值。

axis auto 设置系统到它的缺省动作——自动计算当前轴的范围，取决于输入参量 x、y 与 z 的数据中的最大值与最小值。同时将当前坐标轴的属性 XLimMode、YLimMode 与 ZLimMode 设置为“auto”，用户可以指定对某一坐标轴进行自动操作，例如

axis 'auto x'　自动计算 x 轴的范围。

axis 'auto yz'　自动计算 y 轴与 z 轴的范围。

axis manual、axis(axis) 把坐标固定在当前的范围，这样，若保持状态(hold)为 on，后面的图形仍用相同界限，该命令设置了属性 XLimMode、YLimMode 与 ZlimMode 为 manual。

表 4-4 显示由上面三个命令设置的坐标轴属性。

表 4-4　坐标轴属性表

命令坐标轴属性	axis equal	axis normal	axis square	axis tightequal
DataAspectRatioMode	[1 1 1]	没有设置	没有设置	[1 1 1]
PlotBoxAspectRatio	manual	auto	auto	manual
PlotBoxAspectRatioMode	[3 4 4]	没有设置	[1 1 1]	Auto
Stretch-to-fill	禁止	可行	禁止	禁止

axis off 关闭所用坐标轴上的标记、格栅和单位标记，但保留由 text 和 gtext 设置的对象。axis on 显示坐标轴上的标记、单位和格栅。[mode,visibility,direction] = axis('state') 返回表明当前坐标轴的设置属性的三个字符串，如表 4-5 所示。

表 4-5　坐标轴的属性设置

输出参量	返回字符串	说　　明
Mode	auto 或 manual	若 XLimMode、YLimMode 与 ZLimMode 都设置为 auto，则 mode 为 auto；若 XLimMode、YLimMode 或者 ZLimMode 都设置为 manual，则 mode 为 manual
Visibility	on 或 off	
Direction	xy 或 ij	

【例 4-39】 axis 函数举例。

```
>>x = 0:.025:pi/2;
>>plot(x,exp(x).*sin(2*x),'-m<')
>>axis([0  pi/2  0  5])
```

图形结果如图 4-39 所示。

2. Hidden

功能：在一网格图中显示隐含线条。隐含线条的显示，实际上是显示那些从观察角度看没有被其他物体遮住的线条。

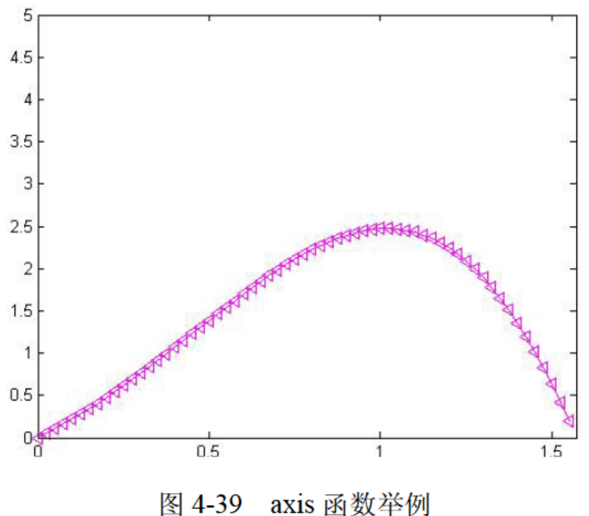


图 4-39　axis 函数举例

用法：hidden on 对当前图形打开隐含线条的显示状态，使网格图后面的线条被前面的线条遮住。设置曲面图形对象的属性 FaceColor 为坐标轴背景颜色，这是系统的缺省操作。hidden off 对当前图形关闭隐含线条的显示，hidden 可在 on 与 off 两种状态之间切换。

【例 4-40】hidden 函数举例。

```
>>mesh(peaks)
>>hidden off
```

图形结果如图 4-40 所示。

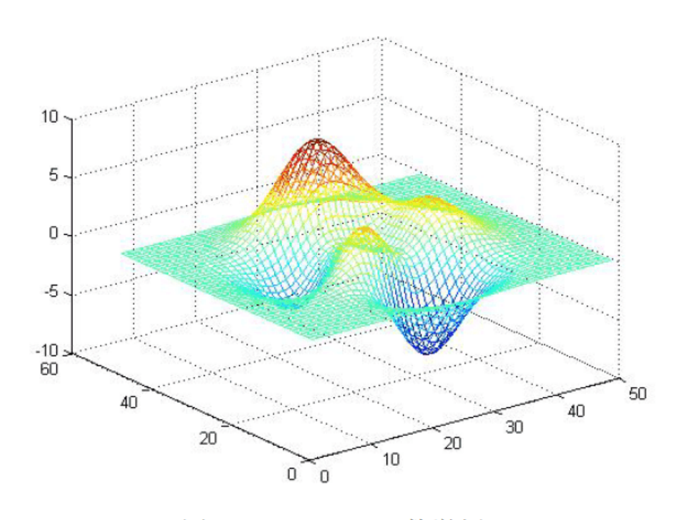


图 4-40　hidden 函数举例

3. shading

功能：设置颜色色调属性，该命令控制曲面与补片等的图形对象的颜色色调，同时设置当前坐标轴中的所有曲面与补片图形对象的属性 EdgeColor 与 FaceColor。命令 shading 设置恰当的属性值，这取决于曲面或补片对象是表现网格图还是实曲面。

用法：shading flat 使网格图上的每一线段与每一小面有相同颜色，该颜色由线段末端的端点颜色确定；或由小面的、有小型的下标或索引的四个角的颜色确定。 shading faceted 是带重叠的黑色网格线的平面色调模式，这是缺省的色调模式。shading interp 在每一线段

与曲面上显示不同的颜色，该颜色为通过在每一线段两边的、或者为不同小曲面之间的色图的索引或真颜色进行内插值得到的颜色。

【例 4-41】 shading 函数举例。

```
>>sphere(16)
>>axis square
>>shading flat
>>title('Flat Shading')
```

图形结果如图 4-41 所示。

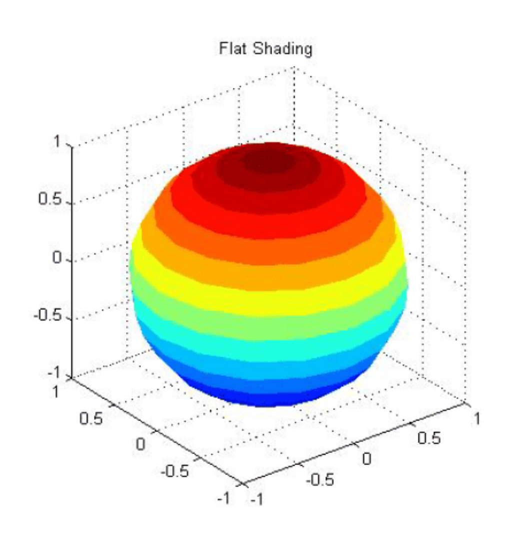


图 4-41 shading 函数举例

4. caxis

功能：颜色坐标轴刻度。命令 caxis 控制着对应色图的数据值的映射图。它影响下面对象之一的、用带索引的颜色数据（CData）与颜色数据映射（CDataMapping）控制的刻度图形对象 surface、patches 与 images；它不影响用颜色数据（CData）或颜色数据映射（CDataMapping）直接设置颜色的图形对象 surface、images 或 patches，该命令还改变坐标轴图形对象的属性 Clim 与 ClimMode。

用法：caxis([cmin cmax]) 用指定的最大值与最小值设置颜色范围。数据值中小于 cmin 或大于 cmax 的，将分别映射于 cmin 与 cmax；处于 cmin 与 cmax 之间的数据将线性地映射当前色图。

caxis auto 让系统自动计算数据的最大值与最小值对应的颜色范围，这是系统的缺省动作。数据中的正无穷大（Inf）对应于最大颜色值；负无穷大（-Inf）对应于最小颜色值；带颜色值设置为 NaN 的面或边界将不显示。

caxis manual、caxis(caxis) 冻结当前颜色坐标轴的刻度范围。当 hold 设置为 on 时，可使后面的图形命令使用相同的颜色范围。

v = caxis 返回包含当前正在使用的颜色范围的二维向量 v=[cmin cmax]。

caxis(axes_handle,…) 使用由参量 axis_handle 指定的坐标轴，而非当前坐标轴。

颜色坐标轴刻度工作原理：

使用带索引的颜色数据（CData）与颜色数据映射（CDataMapping）的图形对象 surface、

patches 与 images 将被设置成刻度化的，在每次图形渲染时，将映射颜色数据值设为当前图形的颜色。当颜色数据值等于或小于 cmin 时，将它映射为当前色图中的第一个颜色；当颜色数据值等于或大于 cmax 时，将它映射为当前色图中的最后一个颜色；对于处于 cmin 与 cmax 之间的颜色数据（如 c），系统将执行下列线性转换，以获得对应当前色图（它的长度为 m）中的颜色的索引（当前色图的行指标 index）：index = fix((C-min)/(cmax-cmin)*m)+1。

【例 4-42】 caxis 函数举例。

```
>>[X,Y,Z] = sphere;
>>C = Z;surf(X,Y,Z,C)
>>caxis([-1 3])
```

图形结果如图 4-42 所示。

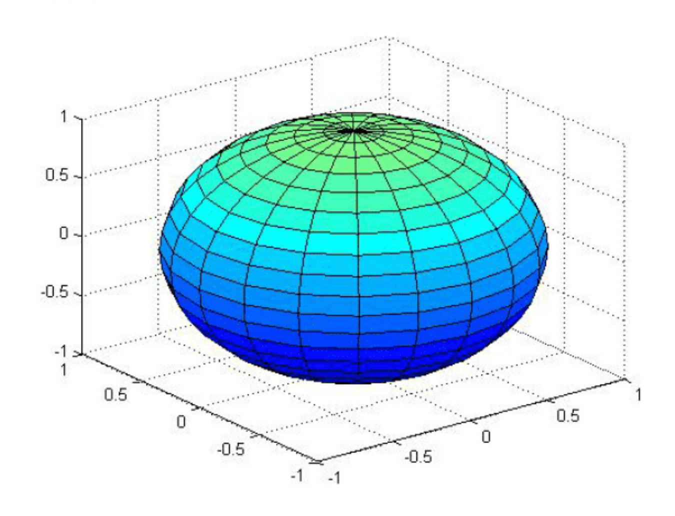


图 4-42 caxis 函数举例

5. view

功能：指定立体图形的观察点。观察者（观察点）的位置决定了坐标轴的方向。用户可以用方位角（azimuth）和仰角（elevation）一起，或者用空间中的一点来确定观察点的位置。

用法：view(az,el)、view([az,el]) 给三维空间图形设置观察点的方位角。方位角 az 与仰角 el 为这两个旋转角度：做一通过视点与 z 轴的平面，与 xy 平面有一交线，该交线与 y 轴的反方向的、按逆时针方向（从 z 轴的方向观察）计算的、单位为度的夹角，就是观察点的方位角 az。若角度为负值，则按顺时针方向计算；在通过视点与 z 轴的平面上，用一直线连接视点与坐标原点，该直线与 xy 平面的夹角就是观察点的仰角 el。若仰角为负值，则观察点转移到曲面下面。

view([x,y,z]) 在笛卡儿坐标系中于点(x,y,z)设置视点。

注意：输入参量只能是方括号的向量形式，而非数学中点的形式。view(2) 设置缺省的二维形式视点。其中，az=0，el=90，即从 z 轴上方观看。view(3) 设置缺省的三维形式视点。其中，az=-37.5，el=30。view(T) 根据转换矩阵 T 设置视点。其中，T 为 4×4 阶的矩阵，如同用命令 viewmtx 生成的透视转换矩阵一样。[az,el] = view 返回当前的方位角 az 与仰角 el。T = view 返回当前的 4×4 阶的转换矩阵 T。

【例 4-43】 view 函数举例。

```
>>peaks;
>>az = 0;el = 90;
>>view(az, el)
```

图形结果如图 4-43 所示。

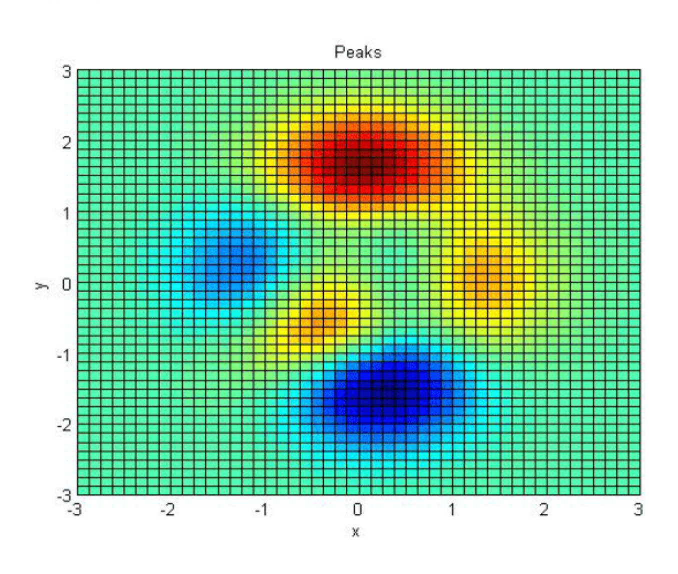


图 4-43　view 函数举例

6．viewmtx

功能：视点转换矩阵。计算一个 4×4 阶的正交的或透视的转换矩阵，该矩阵将一个四维的、齐次的向量转换到一个二维的视平面上（如计算机平面上）。

用法：T = viewmtx(az,el) 返回一个与视点的方位角 az 与仰角 el（单位都是维度）对应的正交矩阵，并没有改变当前视点。

T = viewmtx(az,el,phi) 返回一个透视的转换矩阵，其中，参量 phi 是单位维度的透视角度，为标准化立方体（单位维度）的对象视角角度与透视扭曲程度。表 4-6 对 phi 的值进行了说明。

表 4-6　phi 值说明

Phi 的值	说　　明
0 度	正交投影
10 度	类似以远距离投影
25 度	类似以普通投影
60 度	类似以广角投影

用户可以通过使用返回的矩阵，用命令 view(T)改变视点的位置，该 4×4 阶的矩阵将四维的、同次的向量变换成形式为(x,y,z,w)的非标准化的向量，其中，w 不等于 1。正交化的 x 元素与 y 元素组成的向量(x/w,y/w,z/w,1)为所需的二维向量。（注意：一个四维同次向量为在对应的三维向量后面增加一个 1。例如，[x,y,z,1]为对应于三维空间中的点[x,y,z]的四维向量。）

T = viewmtx(az,el,phi,xc) 返回以在标准化的图形立方体中的点 xc 为目标点的透视矩阵（就像相机正对着点 xc 一样），目标点 xc 为视角的中心点。用户可以用一个三维向量 xc=[xc,yc,zc]指定该中心点，每一分量都在区间[0,1]上。缺省值为 xc=[0 0 0]。

7．Surfnorm

功能：计算与显示三维曲面的法线，该命令计算用户命令 surf 中的曲面法线。

用法：surfnorm(Z)、surfnorm(X,Y,Z) 画出一个曲面与它的法线图。其中，矩阵 Z 用于指定曲面的高度值；X 与 Y 为向量或矩阵，用于定义曲面的 x 与 y 部分。

[Nx,Ny,Nz] = surform(...) 返回组成曲面的法线在三个坐标轴上的投影分量 Nx、Ny 与 Nz。

【例 4-44】 surfnorm 函数举例。

```
>>[x,y,z] = cylinder(1:10);
>>surfnorm(y,x,z)
>>axis([-12 12 -12 12 -0.1 1])
```

图形结果如图 4-44 所示。

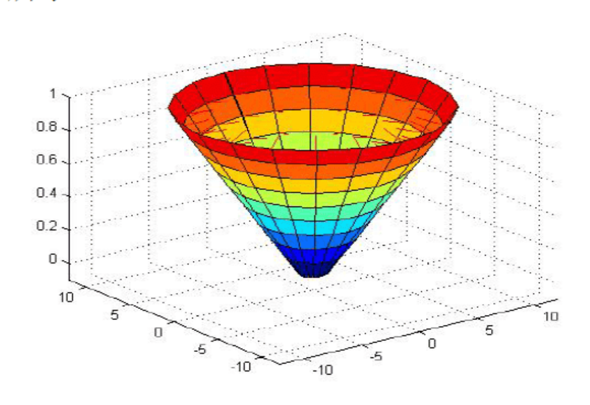


图 4-44 surfnorm 函数举例

4.3.2 文字标示

MATLAB 提供了标题及坐标轴标示和文本标示的文字标示方式，利用这些函数可以为图形加标题，为图形的坐标轴加标注，为图形加图例，也可以把说明、注释文本放到图形的任何位置。函数及其功能如表 4-7 所示。

函　　数	函 数 功 能
title	为图形添加标题
xlable	为 x 轴添加标注
ylable	为 y 轴添加标注
zlable	为 z 轴添加标注
legend	为图形添加图例
text	在指定位置添加文本
otext	用鼠标在图形上放置文本

1．标题和坐标轴标示

title 属性：本坐标轴标题的句柄，其具体内容由 title()函数设定，由此句柄就可以访问到原来的标题。



xlabel 属性：x 轴标注的句柄，其内容由 xlabel()函数设定。此外，类似还有 ylabel 和 zlabel 属性等。xdir 属性，x 轴方向，可以选择“normal”（正向）和“rev”（逆向）。

xgrid 属性：表示 x 轴是否加网格线，可选值为“off”和“on”。

xlim 属性：x 轴上下限，以向量[xm,xM]给出，此外，还有 ylim 和 zlim 属性。

xscale 属性：x 轴刻度类型设置，可以为“linear”（线性的）和“log”（对数的）。此外，还有 yscale 和 zscale 属性。xtick 和 xtickLabel 属性。xtick 属性将给出 x 轴上标尺点值的向量，而 xticklabel 将存放这些标尺点上的标记字符串。对 y 轴和 z 轴也有相应的标尺属性，如 ztick 等。

2. 文本标示

用法：text(x,y,'string')在图形中指定的位置(x,y)上显示字符串 string。

text(x,y,z,'string') 在三维图形空间中的指定位置(x,y,z)上显示字符串 string。

text(x,y,z, 'string'.'PropertyName',PropertyValue…) 对引号中的文字 string 定位于用坐标轴指定的位置，且对指定的属性进行设置。

3. 特殊字符标注

利用 LaTeX 字符集和 MATLAB 文本注释的定义，可以在 MATLAB 的图形文本标注中使用希腊字符、数学符号或上标及下标字体等。

进行上标文本的注释需要使用“^”字符，进行下标文本的注释需要使用“_”字符。

^{supeerstring}——进行上标文本的注释。

_{substring}——进行下标文本的注释。

使用特殊字符标注时，要用“\”符号。

\bf：加粗字体。

\it：斜字体。

\sl：斜字体。

\rm：正常字体。

\fontname{fontname}：定义使用特殊的字体名称。

\fontsize{fontsize}：定义使用特殊的字体大小。

4.3.3 文字修饰

文字标注是图形修饰中的重要因素，它可以是用户在窗口上随意添加的字符说明，还可以是坐标轴对象中所用到的刻度标志等。字符对象的常用属性如下。

Color 属性：字符的颜色，该属性的属性值是一个 1×3 颜色向量。

FontAngle 属性：字体倾斜形式，如正常“normal”和斜体“italic”等。

FontName 属性：字体的名称，如“Times New Roman”与“Courier”等。

FontSize 属性：字号大小，默认以 pt 为单位，属性值应该为实数。

FontWeight 属性：字体是否加黑，可以选择“light”、“normal”（默认值）、“demi”和“bold”4 个选项，其颜色逐渐变黑。

HorizontalAlignment 属性：表示文字的水平对齐方式，可以有“left”（按左边对齐）、“center”（居中对齐）、“right”（按右边对齐）三种选择。

FontUnits 属性：字体大小的单位，如“points”（磅数）为默认的值，此外，还可以使用如下单位“inches”（英寸）、“centimeters”（厘米）、“normalized”（归一值）与“pixels”（像素）等。

Rotation 属性：字体旋转角度，可以为任何数值。

Editing 属性：是否允许交互式修改，选项可以为“on”和“off”。

String 属性：构成本字符对象的字符串，可以是字符串矩阵。

Interpreter 属性：是否允许 TeX 格式，选项为“tex”（允许 TeX 格式）和“none”（不允许）两种，前者显示效果好，后者速度快。

Extent 属性：字符串所在的位置范围，是只读型的，1×4 向量，前两个值表示字符串所在位置的左下角坐标，后两个分量分别为字符对象的长和高。

4.3.4 图例注解及添加颜色条

图例通过对每一条曲线标注不同颜色和应用不同的线条，来区分一张图中绘制的多条曲线。颜色条主要用于显示图形颜色和数值的对应关系，常用于三维图形和二维图形等高线图形中。

1. 图例注解

用户可以通过插入菜单的图例项（legend）为曲线添加图例，也可以使用 legend 函数为曲线添加图例。

当在一个坐标系上画多幅图形时，为区分各个图形，MATLAB 提供了图例的注释说明函数。

其格式如下。

```
legend(字符串 1,字符串 2,字符串 3,…,参数)
```

参数字符串的含义如下表 4-8 所示。

表 4-8 参数字符串的含义

参数字符串	含　义
0	尽量不与数据冲突，自动放置在最佳位置
1	放置在图形的右上角
2	放置在图形的左上角
3	放置在图形的左下角
4	放置在图形的右下角
-1	放置在图形视窗的外右边

此函数在图中开启了一个注释视窗，依据绘图的先后顺序，依据输出字符串对各图形进行注释说明。如字符串 1 表示第一个出现的线条，字符串 2 表示第二个出现的线条，参数字符串确定注释视窗在图形中的位置。同时，注释视窗也可以用鼠标拖动，以便将其放置在一个合适的位置。

【例 4-45】 在同一坐标内，绘出两条函数曲线并有图解注释。

```
>>x=0:0.2:12;
```

```
>>plot(x,sin(x),'-',x,1.5*cos(x),':');
>>legend('First','Second',-1); %强行将注释视窗放在图形视窗的外右边
```

程序运行的结果如图 4-45 所示。

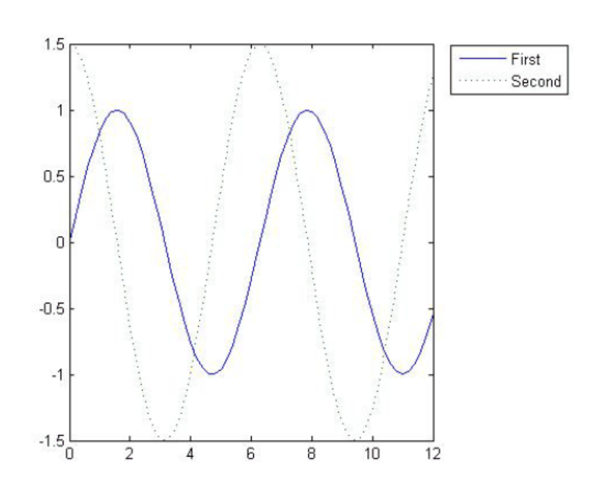


图 4-45　带注释的图

2. 增加颜色条

用户可以通过插入菜单的颜色条项（colorbar）为图形添加颜色条，也可以使用 colorbar 函数为图形添加颜色条。

1）colorbar

在当前坐标轴的右侧添加新的垂直方向的颜色条。如果在那个位置已经存在颜色条，colorbar 函数将使用新的颜色条替代它。如果在非默认的位置存在颜色条，则保留该颜色条。

2）colorbar('off'), colorbar('hide')和 colorbar('delete')

删除所有与当前坐标轴相关联的颜色条。

3）colorbar(...,'peer',axes_handle)

创建与 axes_handle 所代表的坐标轴相关联的颜色条。

4）colorbar(...,'location')

在相对于坐标轴的指定方位添加颜色条，如果在指定的方位存在颜色条，则它将被新的颜色条取代。location 可以是如下值。

North 为图形边框内部靠近上方的位置；South 为图形边框内部靠近下方的位置；East 为图形边框内部靠近右方的位置；West 为图形边框内部靠近左方的位置；NorthOutside 为图像边框外部靠近上方的位置；SouthOutside 为图形边框外部靠近下方的位置；EastOutside 为图形边框外部靠近右方的位置；WestOutside 为图形边框外部靠近左方的位置；使用 Outside 为值来设置 location 能确保颜色条不会覆盖坐标轴中的图形。

5）colorbar(...,'PropertyName',PropertyValue)

指定用来创建颜色条的坐标轴的属性名称和属性值。location 属性值仅适用于颜色条和图例，不适用于坐标轴。

6）cbar_axes = colorbar(...) book.iLoveMatlab.cn

返回新的颜色条对象的句柄，颜色条对象是当前窗口的子对象。如果颜色条已经存在，将创建一个新的颜色条。

7）colorbar(cbar_handle, 'PropertyName',PropertyValue,...)

为 cbar_handle 所代表的颜色条对象设置属性值。要得到已存在的颜色条的句柄，使用如下命令。

```
cbar_handle = findobj(figure_handle,'tag','colorbar')
```

其中，figure_handle 是包含颜色条的图形窗口的句柄。如果图形窗口包含多个颜色条，返回的 cbar_handle 是一个向量，用户需要选择指向要修改的颜色条的句柄。

4.3.5　图形的保持

MATLAB 提供了 hold 命令用来保持当前图形。系统默认的是在当前图形窗口中绘图，如果一个图形绘制完成后，需要继续绘图，系统将原图形覆盖，并在原窗口中绘制图形。要想保持原有图形，并在图形中添加新的内容，就会用到 MATLAB 的保持当前图形的功能。

hold on：保持当前图形。

hold off：解除 hold on 命令。

【例 4-46】 图形执行 hold 命令。

```
>>x=linspace(0,2*pi,30);
>>y=sin(x);
>>plot(x,y)
```

先画好一个图形，然后用下述命令增加 cos(x)的图形。

```
>>hold on
>>z=cos(x); plot(x,z)
>>hold off
```

执行 hold on 后的图形如图 4-46 所示。

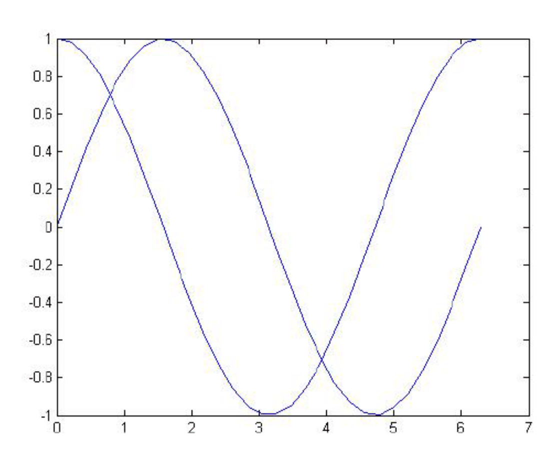


图 4-46　执行 hold on 后的图形

4.3.6　网格控制及坐标轴封闭

MATLAB 提供了控制网格显示和显示的函数，分别为 grid 函数和 box 函数，默认形式

是不划分网格且坐标轴封闭。

MATLAB 提供了 grid 函数用于设置网格线，给坐标加网格线用 grid 命令来控制。grid on/off 命令控制是画还是不画网格线，不带参数的 grid 命令在两种状态之间进行切换。

具体用法如下。

grid：是否划分网格线的切换指令。

grid on：添加网格线。

MATLAB 的绘图确实很强大，只是一直觉得其 grid 控制不灵活。如用 semilogy 绘图，显示 grid 时，一般默认的显示除了 1、0.1、0.01 等的 grid 线外，还会显示 0.2、0.3 这样的 grid 线，尽管在坐标轴上并没有标注。有时这么多 grid 线显得杂乱，若要把 0.2、0.3 的 grid 线去掉，有以下几种方法。

（1）先求对数，再用 plot 绘图，这样的 grid 设置要简单点，或者在图像属性中设置，或者用 set 函数修改属性，比如：

```
set(gca,'ytick',[-4 -3 -2 -1])
```

只是这样需要修改坐标轴的刻度标注，而且似乎没法用上角标表示指数。

（2）mathworks file exchange 上有一个程序 grid2，它扩展了 grid 命令的一些功能，可以对单个坐标轴设置。grid2 minor 显示所有 minor grid，用 grid minor 可以清除所有 minor grid。如果只用 grid minor 可能显示 x 轴的 minor grid 而清除 y 轴的 minor grid，或者相反。

（3）图像窗口的 Property Editor->Property Inspector 对话框可以设置所有的对象属性，相关的有 XMinorTick、XMinorGrid、YMinorTick、YMinorGrid 等，直接修改即可，这与调用 set 函数的效果相同。

【例 4-47】 为图形添加网格线。

```
>>x=linspace(0,2*pi,30);
>>y=sin(x);
>> plot(x,y)
>>grid on
```

添加网格线后如图 4-47 所示。

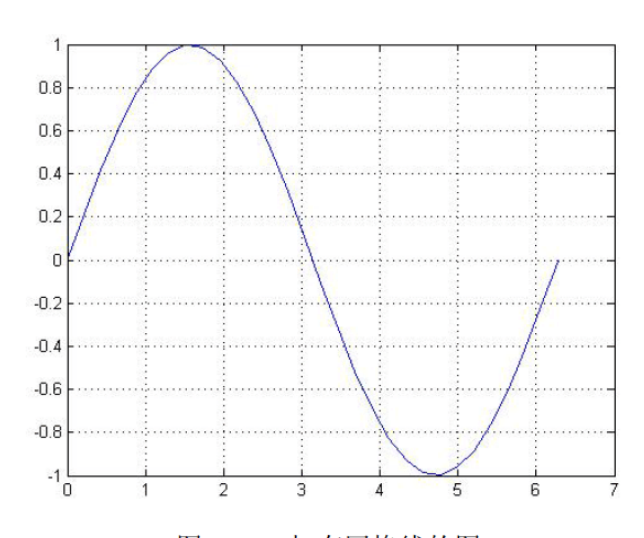


图 4-47　加有网格线的图

grid off：取消网格线。

box 函数：坐标形式在封闭和开启间切换。

box on：坐标呈封闭形式，默认形式。

box off：坐标呈开启形式。

4.3.7 图形窗口的分割

MATLAB 提供了 subplot 函数用于对图形窗口进行分割。subplot 函数的功能是将绘图窗口分割成多个矩形子区域，在指定的子区域绘图，它的具体用法如下所示。

subplot(m,n,p)：将当前绘图窗口分割成 m×n 个子区域，并指定第 p 个编号区域是当前的绘图区域，区域编号的原则是“从上到下，从左到右”。

subplot(m,n,p,'replace')：如果指定区域已存在坐标系，则删掉已有坐标系创建新坐标系。

subplot(m,n,p, 'align')：将坐标系对齐。

subplot(h)：在句柄 h 指定的坐标系中绘图。

subplot('position',[left bottom width height])：在由 4 个元素指定的位置上创建坐标。

【例 4-48】 分割图形窗口。

```
>>y2=sin(15*t)
>>subplot(211)
>>ploy(t,y1)
>>plot(t,y1)
>>subplot(212)
>>plot(t,y2)
```

分割后的图形窗口如图 4-48 所示。

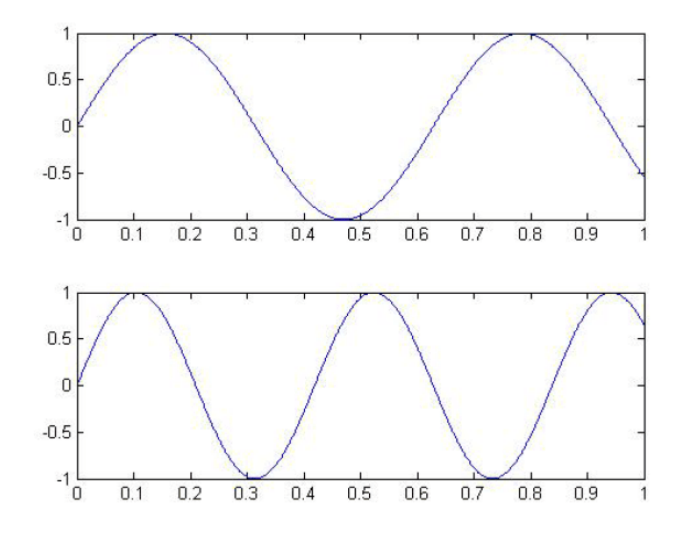


图 4-48 图形窗口的分割

4.4 图形窗口

MATLAB 的图形都是在图形窗口中绘制的，创建图形窗口有两种方式：一种是绘制时，系统自动创建图形窗口，另外一种是采用创建图形函数来创建图形窗口。图形窗口包含菜单栏和工具栏，用户可以通过这些对图形对象进行操作。

4.4.1 图形窗口的创建与控制

1. 图形窗口的创建

MATLAB 提供了 figure 函数用于创建图形窗口，它的具体用法如下：在 MATLAB 下建立一个图形窗口（图 4-35）由命令 figure 完成（或命令窗口 File-New-Figure 选项），每执行一次 figure 就产生一个图形窗口，可以同时产生若干个图形窗口，MATLAB 自动把这些窗口的名字添加序号（No.1，No.2，…）作为区别，同时，这些窗口都被自动分配一个句柄，窗口上有菜单和工具条，其中包括通用的文件操作命令、编辑命令，对图形的坐标轴、线型等特性进行设置的专用工具，还可以为图形添加标注。

说明如下。

（1）MATLAB 在图形窗口中绘制或输出图形，因此图形窗口就像一张绘图纸。

（2）在 MATLAB 下，每一个图形窗口有唯一的一个序号 h，称为该图形窗口的句柄，MATLAB 通过管理图形窗口的句柄来管理图形窗口。

（3）当前窗口句柄可以由 MATLAB 函数 gcf 获得。

（4）在任何时刻，只有唯一的一个窗口是当前的图形窗口（活跃窗口）；figure(h)将句柄为 h 的窗口设置为当前窗口。

在运行绘图程序前若已打开图形窗口，则绘图函数不再打开，而是直接利用已打开的图形窗口；若运行程序前已存在多个图形窗口，并且没有指定哪个窗口为当前窗口时，则以最后使用过的窗口为当前窗口输出图形。

【例 4-49】 创建图形窗口。

```
>> figure
```

创建图形窗口如图 4-49 所示。

figure(h)：如果 h 句柄所对应的窗口对象已存在，则该命令使得该图形窗口成为当前窗口；如果不存在，则新建一个以 h 为句柄的窗口。

h=figure(...)，返回图形窗口对象的句柄。

2. 图形窗口的控制

使用 figure 函数创建图形窗口后，要实现对窗口的控制，可以有两种方法：

一种是使用属性编辑器；另外一种是使用 MATLAB 提供的 get 函数和 set 函数。

3. 关闭图形窗口

关闭图形窗口由 close 命令完成，每执行一次 close 命令关闭一个当前的图形窗口，要同时关闭所有窗口，用 close all 来完成。

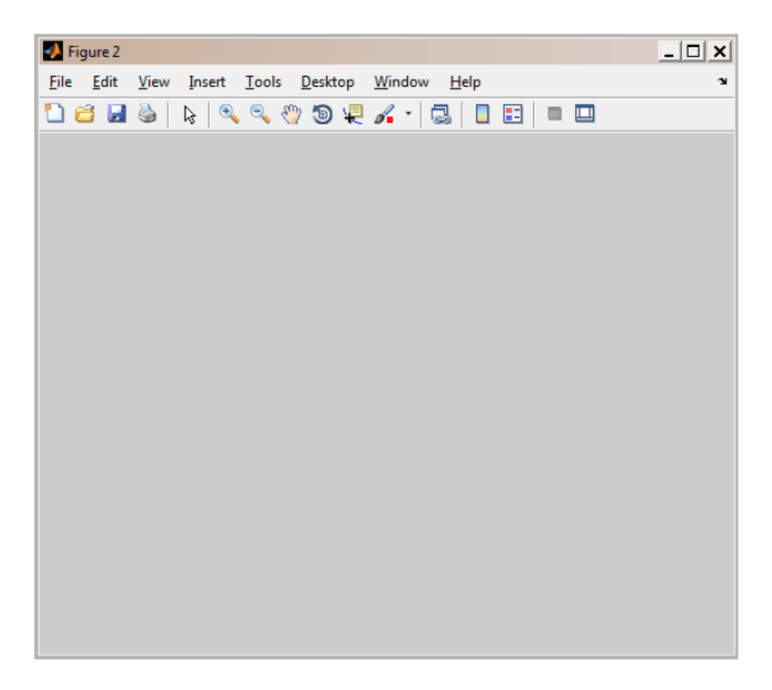


图 4-49　图形窗口

4.4.2　图形窗口的菜单操作

本小节介绍图形窗口中的常用菜单命令。

1．file 菜单

MATLAB 中 File 菜单的命令形式和 Windows 系统中 File 菜单的命令形式类似，包括 New、Open、Close、Save 和 SaveAs 等命令，如图 4-50 所示。

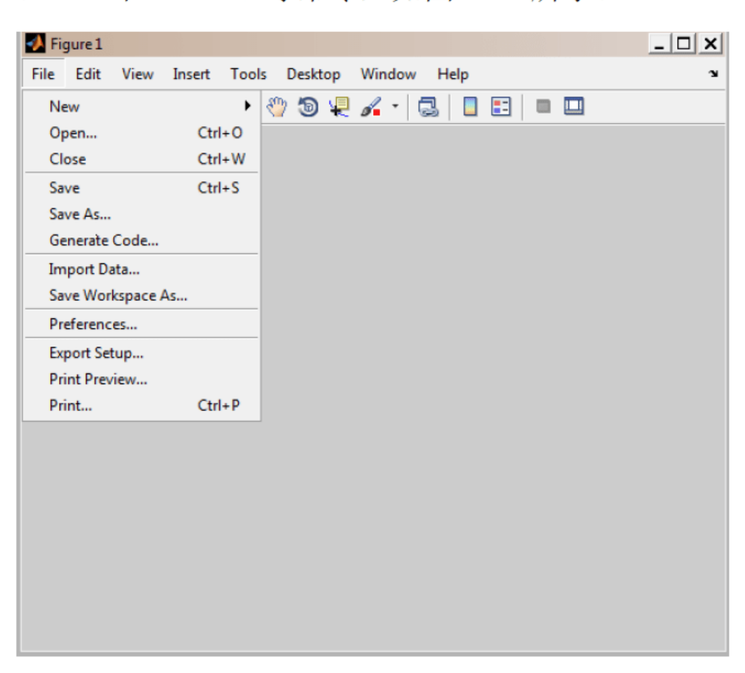


图 4-50　file 菜单命令

New 命令有三个选项。选择“M-file”，表示新建一个 M 文件，该命令将打开 MATLAB

的 M 文件编辑/调试器。通过 M 文件编辑/调试器，用户可以创建自己的 M 文件，也可以编辑已有的 M 文件并调试 MATLAB 程序。“Figure”选项表示新建一个图形窗口。

Open：打开已经存在的文件。

Save：保存文件。

SaveAs 另存文件。

Generate M-File：生成 M 文件。

2. Edit 菜单

Edit 菜单命令如图 4-51 所示。

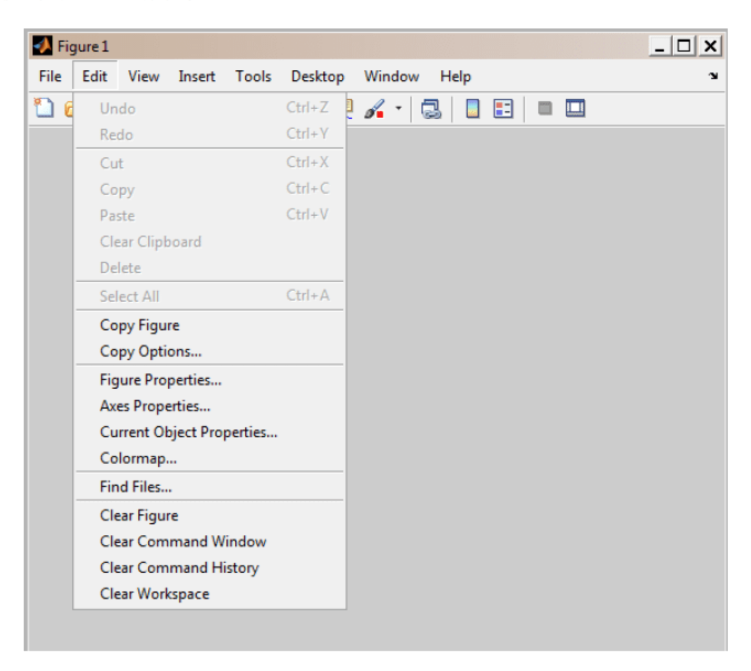


图 4-51　Edit 菜单命令

Copy Figure：用于复制图形。

Copy Option：可以设置图形复制的格式，图形背景颜色和图形大小等。

Figure Properties：可以对图形的属性进行设置，如图形窗口的标题，颜色映射表。单击 More Properties 按钮可以获得更多属性设置。

Axes Properties：用于打开设置坐标轴属性对话框。

Current Object Properties：用于打开设置图形窗口中当前对象（如窗口的坐标轴，图形等）属性对话框。

Colormap：用于打开色图编辑对话框，设置图形的颜色表。

3. Insert 菜单

Insert 菜单主要用于向当前图形中插入各种标注图形，如坐标轴、箭头、标题、直线和图例，如图 4-52 所示。

4. Tools 菜单

Tools 菜单包括一些常用的图形工具，如平移、旋转、缩放和观点控制等，并且 Tools 菜单还提供了两个图形分析工具：Basic Fitting 工具和 Data Statics 工具，用于对图形中的数据进行拟合和分析，如图 4-53 所示。

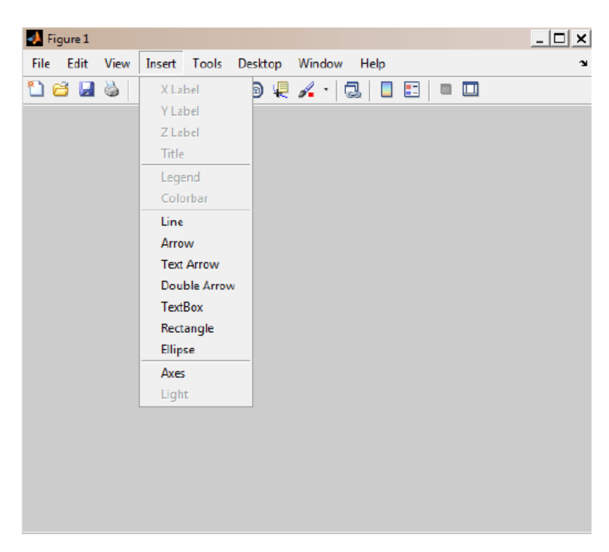


图 4-52　Insert 菜单命令

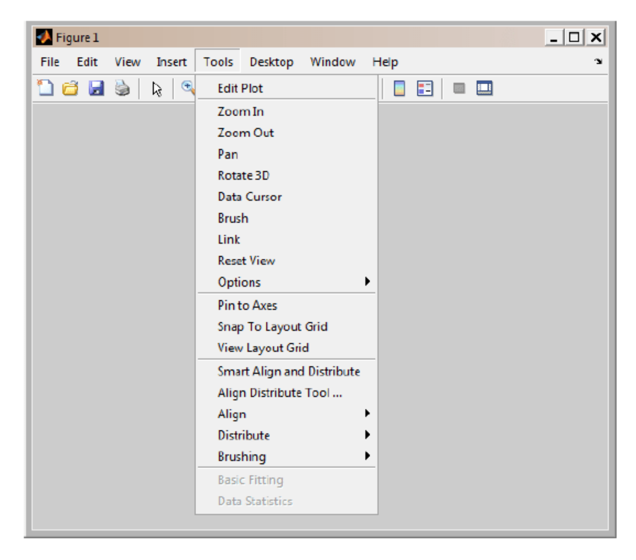


图 4-53　Tools 菜单命令

5. Desktop 菜单

Desktop 菜单用于将窗口合并到 MATLAB 主界面的窗口中，如图 4-54 所示。

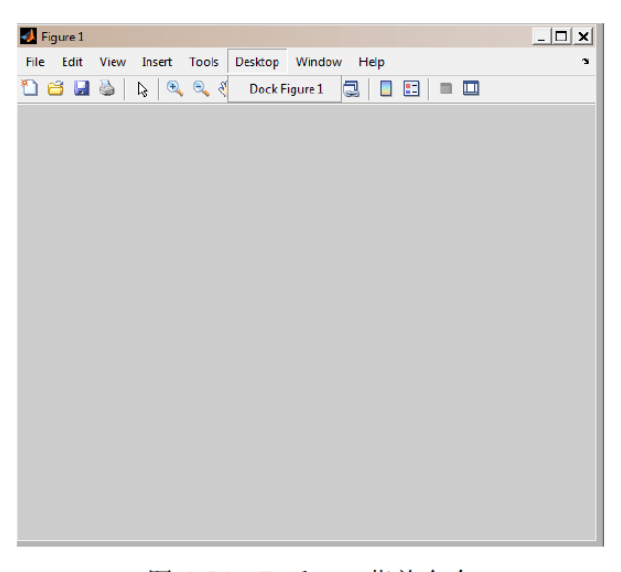


图 4-54　Desktop 菜单命令

单击 Dock Figure1 按钮时显示如图 4-55 所示。

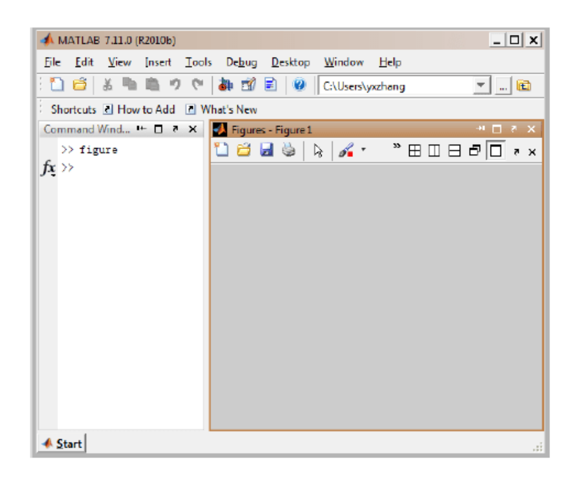


图 4-55　Dock Figure1 命令

4.5 图形文件操作

4.5.1 保存和打开图形文件

MATLAB 提供了一种类似于 MAT 格式的文件用来保存 MATLAB 的图形文件，这种文件的扩展名为*.fig，扩展名为.fig 的图形格式的文件只能在 MATLAB 中使用。

1．第一种方法

保存方法如下

在图形窗体中选择“File”菜单下的“Save”命令，或直接单击工具栏上的保存按钮，在弹出的对话框中选择保存类型为.fig，输入文件名，然后单击“保存”按钮。

打开方式如下。

（1）通过菜单命令或工具栏的按钮可以完成打开操作。

（2）在 MATLAB 的 Current Directory 窗口中双击文件名。

2．第二种方法

使用 saveas 函数保存

```
saveas(h,'filename.ext');
saveas(h,'filename','format');
```

其中，h 为图形的句柄；filename 为保存的文件名；ext 为文件保存的格式；format 为直接说明文件的保存格式。图形文件的扩展名为 m 或 mfig。

打开：使用 open 函数。open 函数根据文件的扩展名不同而调用相应的辅助函数文件 open('filename.ext')。例如：

```
>> surf(peaks(30))
```

将图形文件保存为 M 文件和 fig 文件

```
>> saveas(gcf,'peakfile','M')
```

调用 M 文件重新显示窗体

```
>> peakfile
```

使用 open 指令打开文件

```
>> open('peakfile.fig')
```

4.5.2 导出文件

MATLAB 的图形窗口还可以将图形文件保存成其他的特殊图形格式文件。

MATLAB 支持的图形文件格式。

将图形文件保存成其他的特殊图形格式文件的方法如下。

第一种：执行图形窗体“File”菜单下的“Export”命令，然后在对话框中选择需要导出的图形文件格式，给出文件名，单击“保存”按钮。

第二种：使用 saveas 函数：

```
saveas(h,'filename.ext');
saveas(h,'filename','format');
```

例如，将图形文件保存为 tiff 格式的文件。

```
        saveas(h,'filename.tif');
        saveas(h,'filename','tif');
>> z=peaks(30);
>> surf(z)
>> saveas(gcf,'f','tif')
```

或

```
>> saveas(gcf,'f.tif')
```

第三种：使用 print 函数。

4.6 图像文件操作

4.6.1 打开

为了便于使用，在设计时，通过对话框的形式来选择文件，选择 uigetfile 函数来实现，uigetfile 函数显示一个打开文件对话框，该对话框自动列出当前路径下的目录和文件，由于这个 GUI 程序的操作对象是图像文件，所以设置这里的缺省后缀名为“.bmp”。

【例 4-50】 打开文件对话框。

```
>>uigetfile
```

打开文件对话框如图 4-56 所示。

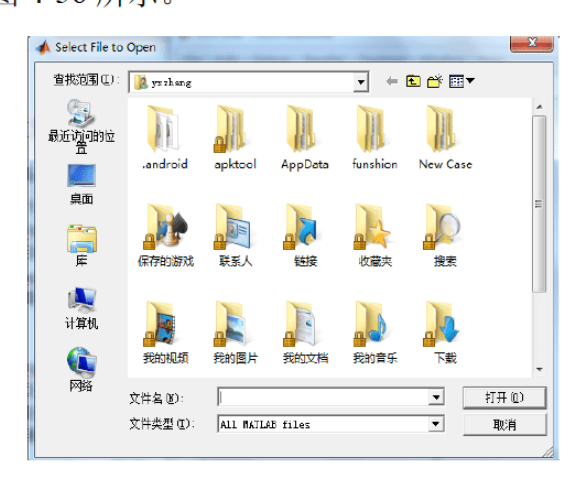


图 4-56　打开文件对话框

uigetfile 函数的调用格式为[name,path]=uigetfile(…)，在按下对话框中的执行按钮“打开”按钮后，返回选择的文件名和路径，分别保存到“name”和“path”中。如果按下取消按钮或是发生错误，则返回值是 0。根据返回值的情况，如果是 0，则弹出提示错误对话框，否则，通过 imread 函数读出图像数据，把图像数据赋值给全局变量 handles.img。

4.6.2 保存

同样也可通过对话框的形式来保存图像数据，通过 uigetfile 函数选择文件名和路径，用 getimage(gca)取出坐标 2 变换后的图像数据保存到变量 i，最后用 imwrite 函数，把数据 i 存到指定的文件。

4.6.3 退出

退出比较简单，程序如下所示。

```
>>clc;
>>close all;
>>close(gcf);
```

4.6.4 图像处理基本操作

本节是 MATLAB 的数字图像处理技术，系统中包括了图像处理技术的各个方面，涵盖了图像处理领域的个别算法，在此过程中所应用的技术和方法为今后的深入研究和将其应用于实际生产奠定了坚实的基础。

MATLAB 对图像的处理功能主要集中在图像处理工具箱(Image Processing Toolbox)中。图像处理工具箱是由一系列支持图像处理操作的函数组成，可以进行诸如几何操作、线性滤波和滤波器设计、图像变换、图像分析与图像增强、二值图像操作及形态学处理等图像处理操作。

1）图像数据的读取

（1）imread：imread 函数用于读入各种图像文件，其一般的用法如下。

```
[X,MAP]=imread('filename','fmt')
```

其中，X，MAP 分别为读出的图像数据和颜色表数据，fmt 为图像的格式，filename 为读取的图像文件（可以加上文件的路径）。

例：[X,MAP]=imread('flowers.tif', 'tif');

（2）imwrite：imwrite 函数用于输出图像，其语法格式为如下。

```
imwrite(X,map,filename,fmt)
```

按照 fmt 指定的格式将图像数据矩阵 X 和调色板 map 写入文件 filename。

读写命令可处理以下图像格式的图像文件，格式文件如表 4-9 所示。

表 4-9　格式文件

格　　式	扩　展　名	相关字符串
微软窗口	bmp	“bmp”
阶层式数据格式	hdf	“hdf”
全彩	Jpg or jpeg	“Jpg” or “jpeg”
微软窗口	pcx	“pcx”
可移植网络图形	png	“png”
标记式图像格式	tiff	“tiff” X
窗口	xwd	“xwd”
图形交换格式	gif	“gif”

（3）imfinfo：imfinfo 函数用于读取图像文件的有关信息，其语法格式为

```
imfinfo(filename,fmt)
```

imfinfo 函数返回一个结构 info，它反映了该图像的各方面信息，其主要数据包括文件名(路径)、文件格式、文件格式版本号、文件的修改时间、文件的大小、文件的长度、文件的宽度、每个像素的位数、图像的类型等。

【例 4-51】 读取图像文件的有关信息。

```
imfinfo('rice.tif')
    ans =
        Filename: 'C:\MATLAB6p5\toolbox\images\imdemos\rice.tif'
        FileModDate: '26-Oct-1996 06:11:58'
        FileSize: 65966
        Format: 'tif'
        FormatVersion: []
        Width: 256
        Height: 256
        BitDepth: 8
        ColorType: 'grayscale'
        FormatSignature: [73 73 42 0]
        ByteOrder: 'little-endian'
        NewSubfileType: 0
        BitsPerSample: 8
        Compression: 'Uncompressed'
   PhotometricInterpretation: 'BlackIsZero'
        StripOffsets: [8x1 double]
        SamplesPerPixel: 1
        RowsPerStrip: 32
        StripByteCounts: [8x1 double]
        XResolution: 72
        YResolution: 72
        ResolutionUnit: 'Inch'
        Colormap: []
       PlanarConfiguration: 'Chunky'
        TileWidth: []
        TileLength: []
```

```
          TileOffsets: []
          TileByteCounts: []
          Orientation: 1
          FillOrder: 1
          GrayResponseUnit: 0.0100
          MaxSamplev  alue: 255
          MinSamplev  alue: 0
          Thresholding: 1
   ImageDescription: [1x166 char]
```

2）MATLAB 中图像文件的显示

imshow 函数可以直接从文件显示多种类型的图像；

image 函数可以将矩阵作为图像显示；

colorbar 函数可以显示颜色条；

montage 函数可以将多幅图像显示在一个图形对象窗口中等。

imshow 函数是最常用的显示各种图像的函数，其语法如下：

```
imshow(X,map)
```

其中 X 是图像数据矩阵，map 是其对应的颜色矩阵，若进行图像处理后不知道图像数据的值域可以用[]代替 map。

需要显示多幅图像时，可以使用 figure 语句，其功能就是重新打开一个图像显示窗口。

【例 4-52】 显示图像文件。

```
I=imread('rice.tif');
Imshow(I);
J=imread('flowers.tif');
figure,imshow(J);
```

3）MATLAB 中灰度直方图的显示

MATLAB 图像处理工具箱提供了 imhist 函数来计算和显示图像的直方图，imhist 函数的语法格式为

```
imhist(I,n)
imhist(X,map)
```

其中，imhist(I,n)计算和显示灰度图像 I 的直方图，n 为指定的灰度级数目，默认值为 256。imhist(X,map)计算和显示索引色图像 X 的直方图，map 为调色板。

```
I = imread('rice.tif');
imshow(I)
figure, imhist(I)
```

4）对比度增强

如果原图像 f(x,y)的灰度范围是[m,M]，希望调整后的图像 g(x,y)的灰度范围是[n,N]，那么下述变换就可以实现这一要求。

MATLAB 图像处理工具箱中提供的 imadjust 函数，可以实现上述的线性变换对比度增

强。imadjust 函数的语法格式为

```
J = imadjust(I,[low_in high_in],[low_out high_out])
```

返回图像 I 经过直方图调整后的图像 J，[low_in high_in]为原图像中要变换的灰度范围，[low_out high_out]指定了变换后的灰度范围。

【例 4-53】 图像对比度的增强。

```
I = imread('pout.tif');
J = imadjust(I,[0.3 0.7],[]);
imshow(I), figure, imshow(J)
```

5）图像的变换

（1）fft2：fft2 函数用于数字图像的二维傅里叶变换，如

```
i=imread('e:\w01.tif');
j=fft2(i);
```

（2）ifft2：ifft2 函数用于数字图像的二维傅里叶反变换，如

```
i=imread('e:\w01.tif');
j=fft2(i);
k=ifft2(j);
```

（3）利用 fft2 函数可以计算二维卷积，如

```
a=[8,1,6;3,5,7;4,9,2];
b=[1,1,1;1,1,1;1,1,1];
a(8,8)=0;
b(8,8)=0;
c=ifft2(fft2(a).*fft2(b));
c=c(1:5,1:5);
```

利用 conv2（二维卷积函数）校验，如

```
a=[8,1,6;3,5,7;4,9,2];
b=[1,1,1;1,1,1;1,1,1];
c=conv2(a,b);
```

6）模拟噪声生成函数和预定义滤波器

（1）imnoise：imnoise 函数用于对图像生成模拟噪声，如

```
i=imread('e:\w01.tif');
j=imnoise(i,'gaussian',0,0.02);%模拟高斯噪声
```

（2）fspecial：fspecial 函数用于产生预定义滤波器，如

```
h=fspecial('sobel');%sobel 水平边缘增强滤波器
h=fspecial('gaussian');%高斯低通滤波器
h=fspecial('laplacian');%拉普拉斯滤波器
```

```
h=fspecial('log');%高斯拉普拉斯(LoG)滤波器
h=fspecial('average');%均值滤波器
```

4.6.5　灰度

由于 RGB 图像是三维图像，所以图像数据是一个三维数组，为了显示灰度图像，把三维图像降为二维，可以只取其中的二维数据，程序为

```
y=(handles.img(:,:,1)); %当然也可以选择(:,:,2) 或(:,:,3)
imshow(y);
```

但是根据所选程序的不同，图像数据也不同，显示也就不同。

另一种方法是运用 rgb2gray 函数实现彩色图像到灰度图像的转换，程序为

```
y=rgb2gray(handles.img);
imshow(y);
```

这个程序只能用于将 RGB 图像转换为灰度图像，当原始图像本来就是灰度图像时，运行该程序就会出错，但是使用者在使用时有时并不知道这些，为了使该程序更加完善，应该在原图像是灰度图像时使用该功能时显示提示类信息。所以在开始时要有一个 RGB 图像或是灰度图像的判断过程，完整的程序如下。

```
if isrgb(handles.img)
y=rgb2gray(handles.img);
imshow(y);
else
   msgbox('这已经是灰度图像','转换失败');
end
```

如果原图是 RGB 图像，执行该操作的结果对比如图 4-57 所示，如果原图本身已经是灰度图像了，执行该操作弹出如图 4-58 所示提示对话框。



（a）处理前



（b）处理后

图 4-57　处理前后的图像

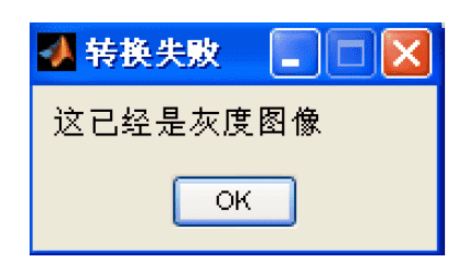


图 4-58　图像本身是灰度图像的结果

4.6.6　截图

在 MATLAB 中，用函数 imcrop 实现对图像的剪切操作，该操作剪切的是图像中的一个矩形子图，用户可以通过参数指定这个矩形四个顶点的坐标，也可以交互地用鼠标选取这个矩形。

Imcrop 函数的调用格式如下。

```
>>y=imcrop(handles.img);
```

不管 handles.img 是三维还是二维数据，该函数都能进行操作。如图 4-59 所示是对三维图像的截图。



（a）处理前



（b）处理后

图 4-59　处理截图前后的图像

4.6.7　缩放

在 MATLAB 中，用函数 imresize 实现对图像的放大或缩小。插值方法可选用三种，最近邻插值、双线性插值和双三次插值。

该函数的调用格式如下。

```
B=imresize(A,m,method)
```

其中，参数 method 用于指定插值的方法，可选的值为 nearest（最近邻法）、bilinear（双线性插值）和 bicubic（双三次插值），缺省值为“nearest”。

B＝imresizee(A,m,method)表示返回原图 A 的 m 倍放大图像（m 小于 1 时实际上是缩小）；

如图 4-60 所示采用邻近插值法放大和缩小图像，参数值保持默认设置。



（a）处理前　　（b）处理后

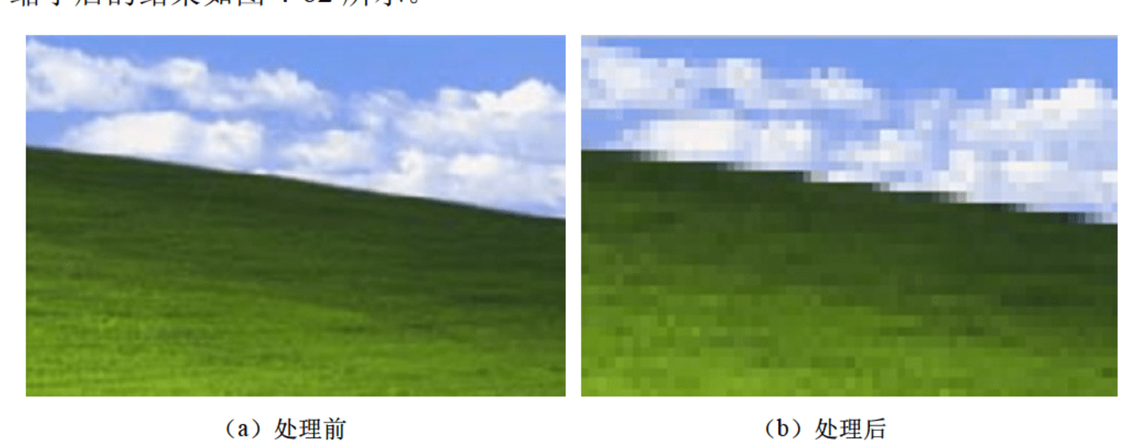
图 4-60　放大前后的图像

虽然处理后看不出放大效果，这是由于坐标轴限制的原因，如果把处理后的图片保存起来，再把处理后的文件打开，就可以看到比较明显的放大效果，如图 4-61 所示。



图 4-61　处理后再次打开的图像

缩小后的结果如图 4-62 所示。



（a）处理前　　（b）处理后

图 4-62　缩小前后的图像

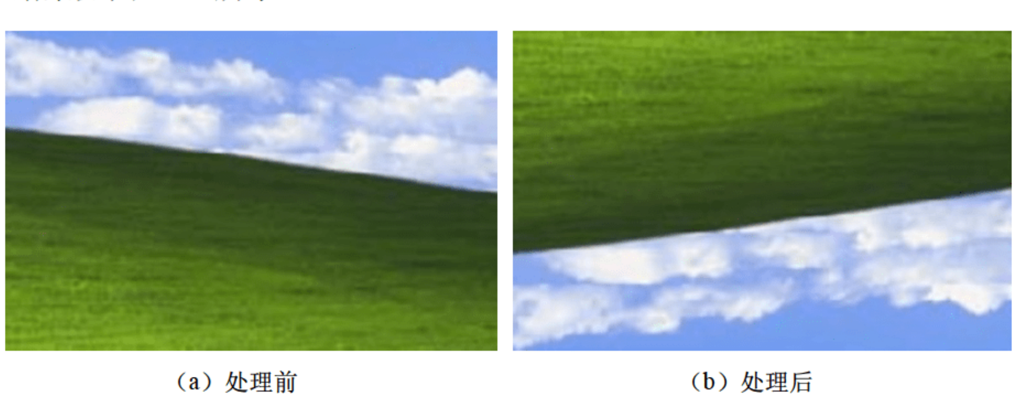
4.6.8 旋转

1. 上下翻转

函数 flipud 实现一个二维矩阵的上下翻转，如 a=[1 2;3 4]，经过该函数处理后，原矩阵变为[3 4;1 2]；所以利用该函数也可以对图像进行上下翻转处理，但由于该函数针对二维数据的处理，所以在写程序时，要对 RGB 图像和灰度图像分开处理，这就要用到 isrgb 函数来判断，如果是灰度图像，可以直接用这个函数进行处理，否则，就要对 RGB 图像进行降维处理。

```
for k=1:3
y(:,:,k)=flipud(x(:,:,k));
end
```

处理结果如图 4-63 所示。

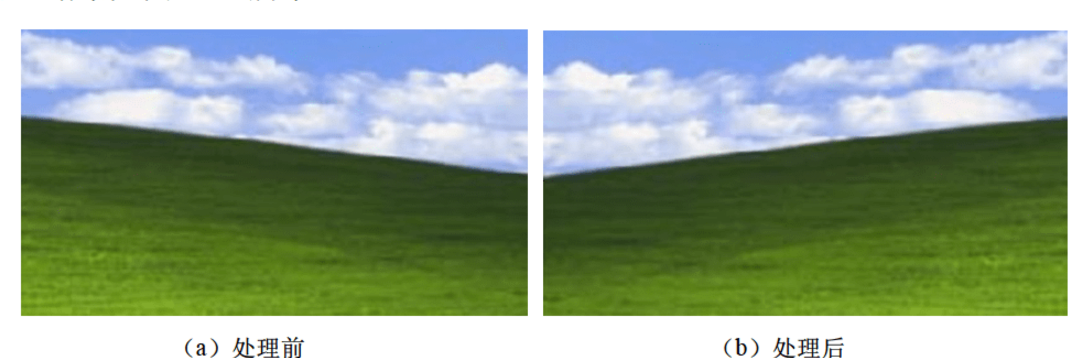


（a）处理前　　（b）处理后

图 4-63　上下翻转前后的图像

2. 左右翻转

对图像的左右翻转也可以用 fliplr 函数来处理，同样，也要对灰度和彩色图像分开处理，处理结果如图 4-64 所示。



（a）处理前　　（b）处理后

图 4-64　左右翻转前后的图像

任意角度翻转用函数 imrotate 来实现对图像的插值旋转。

该函数的调用格式如下。

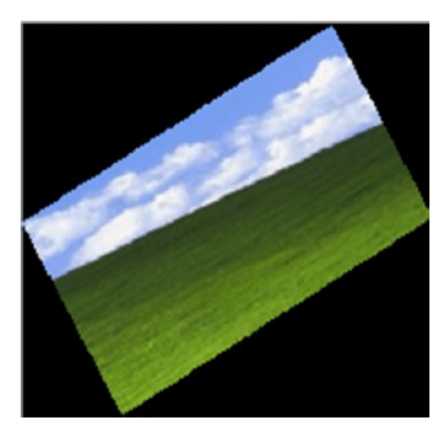
```
B=imrotate(A,angle,method,'crop')
```

其中，参数 method 用于指定插值的方法，可选的值有三种，分别为邻近插值、双线性插值和双三次插值，缺省值为邻近插值，参数 angle 代表旋转的角度。

一般来说，旋转后的图像会比原图大，用户可以指定“crop”参数对旋转后的图像进行剪切（取图像的中间部分），使返回的图像与原图大小相同。执行结果如图 4-65 所示。



（a）处理前



（b）处理后

图 4-65　任意角度翻转前后的图像

4.7　MATLAB 图像分析

MATLAB 的影像处理工具箱支持多种标准的图像处理操作，以方便用户对图像进行分析和调整，这些图像处理操作主要包括。

- 获取像素值及其统计数据。
- 分析图像，抽取其主要结构信息。
- 调整图像，突出其某些特征或抑制噪声。
- 图像质量的分析与处理。

4.7.1　像素及其处理

MATLAB 的影像处理工具箱提供了多个函数以返回与构成图像的数据值相关的信息，这些函数能够以多种显示返回的图像数据的信息。

1．选定像素的数据值（pixval 函数和 impixel 函数）

影像处理工具箱中包含两个函数可以返回用户指定的图像像素的颜色数据值。

1）pixval 函数

当光标在图像上移动时，该函数以交互方式显示像素的数据值。另外，该函数还可以显示两个像素之间的距离。

2）impixel 函数

impixel 函数可以返回选中像素或像素集的数据值。用户可以直接将像素坐标作为该函数输入参数，或者用鼠标选中像素。例如，在下面的例子中，首先调用 impixel 函数，

然后在显示的 come.tif 图像中用鼠标点重（左键选择像素，右键结束），代码如下。

```
Imshow canoe.tif
vals=impixel
```

对于索引图像，pixval 函数和 impixel 函数都将其显示为存储在颜色映像表中，注意是 RGB 值而不是索引值。

2．强度描述图

MATLAB 影像处理工具箱中提供的 improfile 函数用于沿着图像中一条直线段或直线路经路径计算并绘制其强度（灰度）值，代码如下。

```
imshow debye1.tif
improfile
```

执行后，得到运行界面，单击左键确定直线段或直线路径后，按右键，则得到轨迹强度（灰度）图。注意，强度图中的峰值对应于灰度图中的黑色或白色。如果选择任意一条水平线，其自动作出的强度分布图如图 4-66 所示。

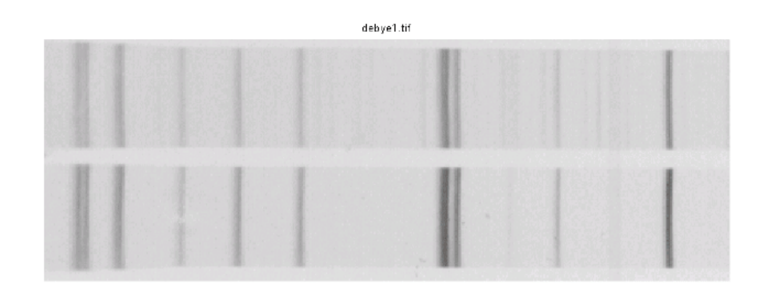


图 4-66　强度分布图

如果选择的是任意一条水平线，其自动作出的强度分布图如图 4-67 所示。

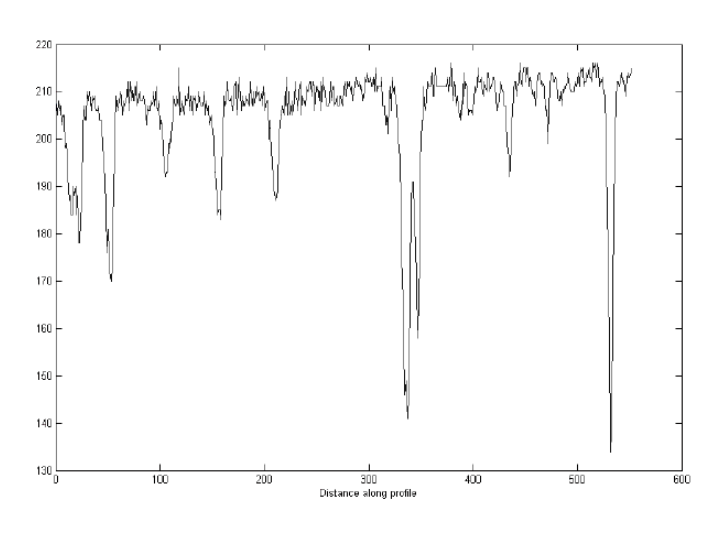
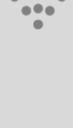


图 4-67　强度分布图

3．图像轮廓图

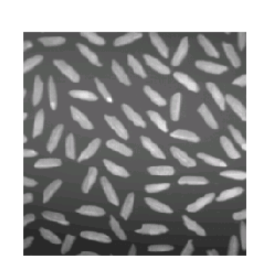
可以利用 Matlab 影像处理工具箱中的 imcontour 数显示灰度图小数据的轮廓图，该函数类似 contour 函数，与 contour 函数相比，功能更全。它能够自动设置坐标轴对象，从



而使得其方向和纵横比能够与要显示的图形相匹配。下面的例子显示一个大米堆的灰度图及其该灰度图像数据的轮廓图，代码如下。

```
I=imread('rice.tif');
subplot(221)
imshow(I)
subplot(222)
imcontour(I)
```

代码执行后，分别用于显示大米灰度图及其轮廓如图 4-68 所示。



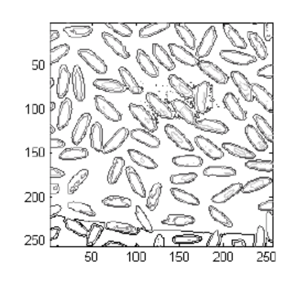


图 4-68　灰度图及其轮廓图

4. 图像柱状图

图像柱状图可以用来显示索引图像或灰度图像中的灰度分布。MATLAB 影像处理工具箱中提供的图像柱状图函数 imhist 可以创建这样的柱状图，还以前面的大米灰度图为例，创建该图的柱状图，代码如下。

```
I=imread('rice.tif');
imhist(I,64)
```

代码执行的结果如图 4-69 所示。从图中可以看出，柱状图的峰值出现在 100 附近，这是因为大米堆的背景色为深灰色所致。

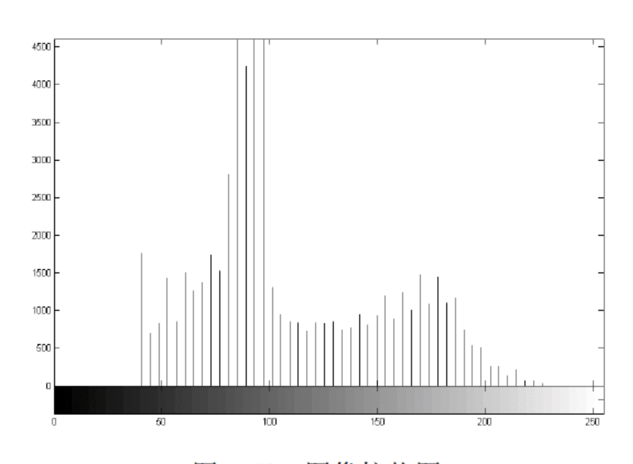


图 4-69　图像柱状图

5. 边界探测器

MATLAB 中的图像分析技术可以提取图像的结构信息。例如，可以利用影像处理工具箱中提供的 edge 函数来探测边界。这里所谓的边界，其实就是图像对象中的边界所对

应的位置，该函数只能应用于灰度图像，其基本原理就是识别图像中灰度值变化较大的像素点。

4.7.2 MATLAB 图像处理工具箱

Image Processing Toolbox（图像处理工具箱）提供一套全方位的参照标准算法和图形工具，用于进行图像处理、分析、可视化和算法开发。可进行图像增强、图像去模糊、特征检测、降噪、图像分割、空间转换和图像配准，该工具箱中的许多功能支持多线程，可发挥多核和多处理器计算机的性能。图像处理工具箱支持多种多样的图像类型，包括高动态范围、千兆像素分辨率、ICC 兼容色彩和断层扫描图像。图形工具可用于探索图像、检查像素区域、调节对比度、创建轮廓或柱状图以及操作感兴趣区域（ROI）。工具箱算法可用于还原退化的图像、检查和测量特征、分析形状和纹理并调节图像的色彩平衡。工具箱中的函数、函数功能和语法如表 4-10～表 4-25 所示。

表 4-10　通用函数

函　　数	功　　能	语　　法
colorbar	显示颜色条	colorbar colorbar(...,'peer',axes_handle) colorbar(axes_handle) colorbar('location') colorbar(...,'PropertyName',PropertyValue) cbar_axes = colorbar(...)
getimage	从坐标轴取得图像数据	A = getimage(h) [x,y,A] = getimage(h) [...,A,flag] = getimage(h) [...] = getimage
image	创建并显示图像对象	image(C) image(x,y,C) image(...,'PropertyName',PropertyValue,...) image('PropertyName',PropertyValue,...) Formal syntax - PN/PV only handle = image(...)
imagesc	按图像显示数据矩阵	imagesc(C) imagesc(x,y,C) imagesc(...,clims) h = imagesc(...)
imshow	显示图像	imshow(I,n) imshow(I,[low high]) imshow(BW) imshow(X,map) imshow(RGB) imshow(…,display_option) imshow(x,y,A,…) imshow filename h = imshow(…)

续表

函　　数	功　　能	语　　法
imview	利用图像浏览器显示图像	imview(I) imview(RGB) imview(X,map) imview(I,range) imview(filename) imview(...,'InitialMagnification',initial_mag) h = imview(...) imview close all
montage	在矩形框中同时显示多帧图像	montage(I) montage(BW) montage(X,map) montage(RGB) h = montage(...)
immovie	创建多帧索引色图像的电影动画	mov = immovie(X,map) mov = immovie(RGB)
subimage	在一个图形中显示多个图像，结合函数 subplot 使用	subimage(X,map) subimage(I) subimage(BW) subimage(RGB) subimage(x,y,...) h = subimage(...)
truesize	调整图像显示尺寸	truesize(fig,[mrows mcols]) truesize(fig)
wrap	将图像显示到纹理映射表面	warp(X,map) warp(I,n) warp(BW) warp(RGB) warp(z,...) warp(x,y,z,...) h = warp(...)
zoom	缩放图像或图形	zoom on zoom off zoom out zoom reset zoom zoom xon zoom yon zoom(factor) zoom(fig, option)

表 4-11　图像文件 I/O 函数

函　　数	功　　能	语　　法
imfinfo	返回图像文件信息	info = imfinfo(filename,fmt) info = imfinfo(filename)

续表

函　　数	功　　能	语　　法
imread	从图像文件中读取图像	A = imread(filename,fmt) [X,map] = imread(filename,fmt) [...] = imread(filename) [...] = imread(URL,...) [...] = imread(...,idx) (CUR, GIF, ICO, and TIFF only) [...] = imread(..., 'PixelRegion', { ROWS, COLS }) (TIFF only) [...] = imread(...,'frames',idx) (GIF only) [...] = imread(...,ref)　(HDF only) [...] = imread(...,'BackgroundColor',BG) (PNG only) [A,map,alpha] = imread(...) (ICO, CUR, and PNG only)
imwrite	把图像写入图像文件中	imwrite(A,filename,fmt) imwrite(X,map,filename,fmt) imwrite(...,filename) imwrite(...,Param1,Val1,Param2,Val2...)

表 4-12　空间变换函数

函　　数	功　　能	语　　法
findbounds	为空间变换寻找输出边界	outbounds = findbounds(TFORM,inbounds)
fliptform	切换空间变换结构的输入和输出	TFLIP = fliptform(T)
imcrop	剪切图像	I2 = imcrop(I) X2 = imcrop(X,map) RGB2 = imcrop(RGB) I2 = imcrop(I,rect) X2 = imcrop(X,map,rect) RGB2 = imcrop(RGB,rect) [...] = imcrop(x,y,...) [A,rect] = imcrop(...) [x,y,A,rect] = imcrop(...)
imresize	图像缩放	B = imresize(A,m) B = imresize(A,m,method) B = imresize(A,[mrows ncols],method) B = imresize(...,method,n) B = imresize(...,method,h)
imrotate	图像旋转	B = imrotate(A,angle) B = imrotate(A,angle,method) B = imrotate(A,angle,method,bbox)
interp2	2-D 数据插值	ZI = interp2(X,Y,Z,XI,YI) ZI = interp2(Z,XI,YI) ZI = interp2(Z,ntimes) ZI = interp2(X,Y,Z,XI,YI,method)

续表

函　　数	功　　能	语　　法
imtransform	对图像进行二维空间变换	B = imtransform(A,TFORM) B = imtransform(A,TFORM,INTERP) [B,XDATA,YDATA] = imtransform(...) [B,XDATA,YDATA] = imtransform(..., param1, val1, param2, val2,...)
makeresampler	生成重采样结构	R = makeresampler(interpolant,padmethod)
maketform	生成几何变换结构	T = maketform(transformtype,...)
tformarray	多维数组的空间变换	B = tformarray(A, T, R, TDIMS_A, TDIMS_B, TSIZE_B, TMAP_B,F)
tformfwd	正向空间变换	[X,Y] = tformfwd(T,U,V) [X1,X2,X3,...] = tformfwd(T,U1,U2,U3,...) X = tformfwd(T,U) [X1,X2,X3,...] = tformfwd(T,U) X = tformfwd(T,U1,U2,U3,...)
tforminv	逆向空间变换	U = tforminv(X,T)

表 4-13　像素和统计处理函数

函　　数	功　　能	语　　法
corr2	计算两个矩阵的相关系数	r = corr2(A,B)
imcontour	创建图像的轮廓图	imcontour(I) imcontour(I,n) imcontour(I,v) imcontour(x,y,...) imcontour(...,LineSpec) [C,h] = imcontour(...)
imhist	显示图像的直方图	imhist(I,n) imhist(X,map) [counts,x] = imhist(...)
impixel	确定像素颜色值	P = impixel(I) P = impixel(X,map) P = impixel(RGB) P = impixel(I,c,r) P = impixel(X,map,c,r) P = impixel(RGB,c,r) [c,r,P] = impixel(...) P = impixel(x,y,I,xi,yi) P = impixel(x,y,X,map,xi,yi) P = impixel(x,y,RGB,xi,yi) [xi,yi,P] = impixel(x,y,...)
improfile	沿线段计算剖面图的像素值	c = improfile c = improfile(n) c = improfile(I,xi,yi) c = improfile(I,xi,yi,n) [cx,cy,c] = improfile(...) [cx,cy,c,xi,yi] = improfile(...) [...] = improfile(x,y,I,xi,yi) [...] = improfile(x,y,I,xi,yi,n) [...] = improfile(...,method)

续表

函　　数	功　　能	语　　法
mean2	求矩阵元素平均值	B = mean2(A)
pixval	显示图像像素信息	pixval on pixval off pixval pixval(fig,option) pixval(ax,option) pixval(H,option)
regionprops	得到图像区域属性	STATS = regionprops(L,properties)
std2	计算矩阵元素的标准偏移	b = std2(A)

表 4-14　图像分析函数

函　　数	功　　能	语　　法
edge	识别灰度图像中的边界	BW = edge(I,'sobel') BW = edge(I,'sobel',thresh) BW = edge(I,'sobel',thresh,direction) [BW,thresh] = edge(I,'sobel',...) BW = edge(I,'prewitt') BW = edge(I,'prewitt',thresh) BW = edge(I,'prewitt',thresh,direction) [BW,thresh] = edge(I,'prewitt',...) BW = edge(I,'roberts') BW = edge(I,'roberts',thresh) [BW,thresh] = edge(I,'roberts',...) BW = edge(I,'log') BW = edge(I,'log',thresh) BW = edge(I,'log',thresh,sigma) [BW,threshold] = edge(I,'log',...)
qtdecomp	执行四叉树分解	S = qtdecomp(I) S = qtdecomp(I,threshold) S = qtdecomp(I,threshold,mindim) S = qtdecomp(I,threshold,[mindim maxdim]) S = qtdecomp(I,fun) S = qtdecomp(I,fun,P1,P2,...)
qtgetblk	获取四叉树分解中的数组块值	[vals,r,c] = qtgetblk(I,S,dim) [vals,idx] = qtgetblk(I,S,dim)
qtsetblk	设置四叉树分解中的数组块值	J = qtsetblk(I,S,dim,vals)

表 4-15　图像增强函数

函　　数	功　　能	语　　法
adapthisteq	执行对比度受限的直方图均衡	J = adapthisteq(I) J = adapthisteq(I,param1,val1,param2,val2...)
decorrstretch	对多通道图像应用解卷积延拓	S = decorrstretch(I) S = decorrstretch(I,TOL)

续表

函　　数	功　　能	语　　法
histeq	用直方图均等化增强对比度	J = histeq(I,hgram) J = histeq(I,n) [J,T] = histeq(I,...) newmap = histeq(X,map,hgram) newmap = histeq(X,map) [newmap,T] = histeq(X,...)
imadjust	调整图像灰度值或颜色映射表	J = imadjust(I) J = imadjust(I,[low_in; high_in],[low_out; high_out]) J = imadjust(...,gamma) newmap = imadjust(map, [low_in high_in], [low_out high_out], gamma) RGB2 = imadjust(RGB1,...)
imnoise	向图像中加入噪声	J = imnoise(I,type) J = imnoise(I,type,parameters)
medfilt2	进行二维中值滤波	B = medfilt2(A,[m n]) B = medfilt2(A) B = medfilt2(A,'indexed',...)
ordfilt2	进行二维统计顺序滤波	B = ordfilt2(A,order,domain) B = ordfilt2(A,order,domain,S) B = ordfilt2(...,padopt)
stretchlim	得到图像对比度延拓的灰度上下限	LOW_HIGH = stretchlim(I,TOL) LOW_HIGH = stretchlim(RGB,TOL)
wiener2	进行二维适应性去噪滤波	J = wiener2(I,[m n],noise) [J,noise] = wiener2(I,[m n])

表 4-16　线性滤波函数

函　　数	功　　能	语　　法
conv2	二维卷积	C = conv2(A,B) C = conv2(hcol,hrow,A) C = conv2(...,'shape')
convmtx2	二维矩阵卷积	T = convmtx2(H,m,n) T = convmtx2(H,[m n])
convn	n 维卷积	C = convn(A,B) C = convn(A,B,'shape')
filter2	二维线性滤波	Y = filter2(h,X) Y = filter2(h,X,shape)
fspecial	创建预定义滤波器	h = fspecial(type) h = fspecial(type,parameters)
imfilter	多维图像滤波	B = imfilter(A,H) B = imfilter(A,H,option1,option2,...)

表 4-17　线性二维滤波器设计函数

函　　数	功　　能	语　　法
freqspace	确定二维频率响应的频率空间	[f1,f2] = freqspace(n) [f1,f2] = freqspace([m n]) [x1,y1] = freqspace(...,'meshgrid') f = freqspace(N) f = freqspace(N,'whole')
freqz2	计算二维频率响应	[H,f1,f2] = freqz2(h,n1,n2) [H,f1,f2] = freqz2(h,[n2 n1]) [H,f1,f2] = freqz2(h) [H,f1,f2] = freqz2(h,f1,f2) [...] = freqz2(h,...,[dx dy]) [...] = freqz2(h,...,dx) freqz2(...)
fsamp2	用频率采样法设计二维 FIR 滤波器	h = fsamp2(Hd) h = fsamp2(f1,f2,Hd,[m n])
ftrans2	通过频率转换法设计二维 FIR 滤波器	h = ftrans2(b,t) h = ftrans2(b)
fwind1	用一维窗口方法设计二维 FIR 滤波器	h = fwind1(Hd,win) h = fwind1(Hd,win1,win2) h = fwind1(f1,f2,Hd,...)
fwind2	用二维窗口方法设计二维 FIR 滤波器	h = fwind2(Hd,win) h = fwind2(f1,f2,Hd,win)

表 4-18　图像变换函数

函　　数	功　　能	语　　法
dct2	进行二维离散余弦变换	B = dct2(A) B = dct2(A,m,n) B = dct2(A,[m n])
dctmtx	计算离散余弦变换矩阵	D = dctmtx(n)
fft2	进行二维快速傅里叶变换	Y = fft2(X) Y = fft2(X,m,n)
fftn	进行 n 维快速傅里叶变换	Y = fftn(X) Y = fftn(X,siz)
fftshift	输出快速傅里叶变换的象限	Y = fftshift(X) Y = fftshift(X,dim)
idct2	计算二维逆离散余弦变换	B = idct2(A) B = idct2(A,m,n) B = idct2(A,[m n])
ifft2	计算二维逆快速傅里叶变换	Y = ifft2(X) Y = ifft2(X,m,n) y = ifft2(..., 'nonsymmetric') y = ifft2(..., 'nonsymmetric')
ifftn	计算 n 维逆快速傅里叶变换	Y = ifftn(X) Y = ifftn(X,siz) y = ifftn(..., 'nonsymmetric') y = ifftn(..., 'nonsymmetric')

续表

函　数	功　能	语　法
iradon	逆 Radon 变换	I = iradon(R,theta) I = iradon(R, theta, interp, filter, frequency_scaling, output_size) [I,H] = iradon(...)
phantom	产生一个头部幻影图像	P = phantom(def,n) P = phantom(E,n) [P,E] = phantom(...)
radon	计算 Radon 变换	R=radon(I,theta) [R,xp]=radon(…)
fanbeam	计算扇形投影变换	F = fanbeam(I,D) F = fanbeam(...,param1,val1,param1,val2,...) [F,sensor_positions,fan_rotation_angles] = fanbeam(...)

表 4-19　边沿和块处理函数

函　数	功　能	语　法
bestblk	确定进行块操作的块大小	siz = bestblk([m n],k) [mb,nb] = bestblk([m n],k)
blkproc	实现图像的非重叠(distinct)块操作	B = blkproc(A,[m n],fun) B = blkproc(A,[m n],fun,P1,P2,...) B=blkproc(A,[m n],[mborder nborder], fun, ...) B = blkproc(A,'indexed',...)
col2im	将矩阵的列重新组织到块中	A = col2im(B,[m n],[mm nn], block_type) A = col2im(B,[m n],[mm nn])
colfilt	利用列相关函数进行边沿操作	B = colfilt(A,[m n],block_type,fun) B = colfilt(A,[m n],block_type,fun,P1,P2,...) B = colfilt(A, [m n], [mblock nblock], block_type, fun,...) B = colfilt(A,'indexed',...)
im2col	重调图像块为列	B = im2col(A,[m n],block_type) B = im2col(A,[m n]) B = im2col(A,'indexed',...)
nlfilter	通用滑动邻域操作	B = nlfilter(A,[m n],fun) B = nlfilter(A,[m n],fun,P1,P2,...) B = nlfilter(A,'indexed',...)

表 4-20　图像形态学操作函数

函　数	功　能	语　法
applylut	在二值图像中利用查找表进行邻域操作	A = applylut(BW,LUT)
bwarea	计算二值图像的对象面积	total = bwarea(BW)
bweuler	计算二值图像的欧拉数	eul = bweuler(BW,n)
bwhitmiss	执行二值图像的击中和击不中操作	BW2 = bwhitmiss(BW1,SE1,SE2) BW2 = bwhitmiss(BW1,INTERVAL)

续表

函　数	功　能	语　法
bwlabel	标注二值图像中已连接的部分	L = bwlabel(BW,n) [L,num] = bwlabel(BW,n)
bwmorph	二值图像的通用形态学操作	BW2 = bwmorph(BW,operation) BW2 = bwmorph(BW,operation,n)
bwperim	计算二值图像中对象的周长	BW2 = bwperim(BW1) BW2 = bwperim(BW1,CONN)
bwselect	在二值图像中选择对象	BW2 = bwselect(BW,c,r,n) BW2 = bwselect(BW,n) [BW2,idx] = bwselect(...) BW2 = bwselect(x,y,BW,xi,yi,n) [x,y,BW2,idx,xi,yi] = bwselect(...)
makelut	创建用于 applylut 函数的查找表	lut = makelut(fun,n) lut = makelut(fun,n,P1,P2,...)
bwdist	距离变换	D = bwdist(BW) [D,L] = bwdist(BW) [D,L] = bwdist(BW,METHOD)
imbothat	执行形态学的闭包运算	IM2 = imbothat(IM,SE) IM2 = imbothat(IM,NHOOD)
imclose	图像的闭运算	IM2 = imclose(IM,SE) IM2 = imclose(IM,NHOOD)
imopen	图像的开运算	IM2 = imopen(IM,SE) IM2 = imopen(IM,NHOOD)
imdilate	图像的膨胀	IM2 = imdilate(IM,SE) IM2 = imdilate(IM,NHOOD) IM2 = imdilate(IM,SE,PACKOPT) IM2 = imdilate(...,PADOPT)
imerode	图像的腐蚀	IM2 = imerode(IM,SE) IM2 = imerode(IM,NHOOD) IM2 = imerode(IM,SE,PACKOPT,M) IM2 = imerode(...,PADOPT)
imfill	填充图像区域	BW2 = imfill(BW,locations) BW2 = imfill(BW,'holes') I2 = imfill(I) BW2 = imfill(BW) [BW2 locations] = imfill(BW) BW2 = imfill(BW,locations,CONN) BW2 = imfill(BW,CONN,'holes') I2 = imfill(I,CONN)
imtophat	用开运算后的图像减去原图像	IM2 = imtophat(IM,SE) IM2 = imtophat(IM,NHOOD)
strel	创建形态学结构元素	SE = strel(shape,parameters)

表 4-21　区域处理函数

函　　数	功　　能	语　　法
roicolor	选择感兴趣的颜色区	BW = roicolor(A,low,high) BW = roicolor(A,v)
roifill	在图像的任意区域中进行平滑插补	J = roifill(I,c,r) J = roifill(I) J = roifill(I,BW) [J,BW] = roifill(...) J = roifill(x,y,I,xi,yi) [x,y,J,BW,xi,yi] = roifill(...)
roifilt2	滤波特定区域	J = roifilt2(h,I,BW) J = roifilt2(I,BW,fun) J = roifilt2(I,BW,fun,P1,P2,...)
roipoly	选择一个感兴趣的多边形区域	BW = roipoly(I,c,r) BW = roipoly(I) BW = roipoly(x,y,I,xi,yi) [BW,xi,yi] = roipoly(...) [x,y,BW,xi,yi] = roipoly(...)

表 4-22　图像代数操作

函　　数	功　　能	语　　法
imadd	加运算	Z = imadd(X,Y)
imsubtract	减运算	Z = imsubtract(X,Y)
immultiply	乘运算	Z = immultiply(X,Y)
imdivide	除运算	Z = imdivide(X,Y)

表 4-23　颜色空间转换函数

函　　数	功　　能	语　　法
hsv2rgb	转换 HSV 的值为 RGB 颜色空间	M = hsv2rgb(H)
ntsc2rgb	转换 NTSC 的值为 RGB 颜色空间	rgbmap = ntsc2rgb(yiqmap) RGB = ntsc2rgb(YIQ)
rgb2hsv	转换 RGB 的值为 HSV 颜色空间	cmap = rgb2hsv(M)
rgb2ntsc	转换 RGB 的值为 NTSC 颜色空间	yiqmap = rgb2ntsc(rgbmap) YIQ = rgb2ntsc(RGB)
rgb2ycbcr	转换 RGB 的值为 YCbCr 颜色空间	ycbcrmap = rgb2ycbcr(rgbmap) YCbCr = rgb2ycbcr(RGB)
ycbcr2rgb	转换 YCbCr 的值为 RGB 颜色空间	rgbmap = ycbcr2rgb(ycbcrmap) RGB = ycbcr2rgb(YCBCR)

表 4-24　图像类型和类型转换函数

函　　数	功　　能	语　　法
dither	通过抖动增加外观颜色分辨率，转换图像	X = dither(RGB,map) BW = dither(I)
gray2ind	转换灰度图像为索引色图像	[X,map] = gray2ind(I,n) [X,map] = gray2ind(BW,n)
grayslice	从灰度图像为索引色图像	X = grayslice(I,n) X = grayslice(I,v)

续表

函　数	功　能	语　法
im2bw	转换图像为二值图像	BW = im2bw(I,level) BW = im2bw(X,map,level) BW = im2bw(RGB,level)
im2double	转换图像矩阵为双精度类型	I2 = im2double(I) RGB2 = im2double(RGB) I = im2double(BW) X2 = im2double(X,'indexed')
double	转换数据为双精度类型	double(X)
uint8	转换数据为 8 位无符号整型	I = uint8(X)
im2uint8	转换图像阵列为 8 位为无符号整型	I2 = im2uint8(I) RGB2 = im2uint8(RGB) I = im2uint8(BW) X2 = im2uint8(X,'indexed')
im2uint16	转换图像阵列为 16 位为无符号整型	I2 = im2uint16(I) RGB2 = im2uint16(RGB) I = im2uint16(BW) X2 = im2uint16(X,'indexed')
uint16	转换数据为 16 位无符号整型	I = uint16(X)
ind2gray	转换索引色图像为灰度图像	I = ind2gray(X,map)
ind2rgb	转换索引色图像为 RGB 图像	RGB = ind2rgb(X,map)
isbw	判断是否为二值图像	flag = isbw(A)
isgray	判断是否为灰度图像	flag = isgray(A)
isind	判断是否为索引色图像	flag = isind(A)
isrgb	判断是否为 RGB 图像	flag = isrgb(A)
mat2gray	转换矩阵为灰度图像	I = mat2gray(A,[amin amax]) I = mat2gray(A)
rgb2gray	转换 RGB 图像或颜色映射表为灰度图像	I = rgb2gray(RGB) newmap = rgb2gray(map)
rgb2ind	转换 RGB 图像为索引色图像	[X,map] = rgb2ind(RGB,tol) [X,map] = rgb2ind(RGB,n) X = rgb2ind(RGB,map) [...] = rgb2ind(...,dither_option)

表 4-25　图像复原函数

函　数	功　能	语　法
deconvwnr	用维纳滤波复原图像	J = deconvwnr(I,PSF) J = deconvwnr(I,PSF,NSR) J = deconvwnr(I,PSF,NCORR,ICORR)
deconvreg	用最小约束二乘滤波复原图像	J = deconvreg(I,PSF) J = deconvreg(I,PSF,NOISEPOWER) J = deconvreg(I, PSF, NOISEPOWER, LRANGE) J = deconvreg(I, PSF, NOISEPOWER, LRANGE, REGOP) [J, LAGRA] = deconvreg(I,PSF,...)



续表

函　数	功　能	语　法
deconvlucy	用 Richardson-Lucy 滤波复原图像	J = deconvlucy(I,PSF) J = deconvlucy(I,PSF,NUMIT) J = deconvlucy(I,PSF,NUMIT,DAMPAR) J = deconvlucy(I, PSF, NUMIT, DAMPAR, WEIGHT) J = deconvlucy(I, PSF, NUMIT, DAMPAR, WEIGHT, READOUT) J = deconvlucy(I, PSF, NUMIT, DAMPAR, WEIGHT, READOUT, SUBSMPL)
deconvblind	用盲卷积滤波复原图像	[J,PSF] = deconvblind(I,INITPSF) [J,PSF] = deconvblind(I,INITPSF,NUMIT) [J,PSF] = deconvblind(I, INITPSF, NUMIT, DAMPAR) [J,PSF] = deconvblind(I, INITPSF, NUMIT, DAMPAR, WEIGHT) [J,PSF] = deconvblind(I, INITPSF, NUMIT, DAMPAR, WEIGHT, READOUT) [J,PSF] = deconvblind(..., FUN, P1, P2, ..., PN)

4.7.3　图像处理的常用函数

MATLAB 的图像处理工具箱中有大量的图像处理函数，限于篇幅，不可能逐一介绍，有兴趣的读者可以根据提供的函数信息和需要自己学习和掌握。本节给出部分常用函数，供学习和参考。

1．applylut

功能：在二进制图像中利用 lookup 表进行边沿操作。

语法：

A = applylut(BW,lut)

【例 4-54】 applylut 函数对图像的处理。

```
lut = makelut('sum(x(:)) == 4',2);
BW1 = imread('text.tif');
BW2 = applylut(BW1,lut);
imshow(BW1)
figure, imshow(BW2)
```

处理结果如图 4-70 所示。

相关命令：makelut





图 4-70 applylut 函数结果图

2. bestblk

功能：确定进行块操作的块大小。

语法：

siz = bestblk([m n],k)

[mb,nb] = bestblk([m n],k)

【例 4-55】 bestblk 函数使用。

```
siz = bestblk([640 800],72)
siz =
      64 50
```

相关命令：

blkproc

3. blkproc

功能：实现图像的显式块操作。

语法：

B = blkproc(A,[m n],fun)

B = blkproc(A,[m n],fun,P1,P2,...)

B = blkproc(A,[m n],[mborder nborder],fun,...)

B = blkproc(A,'indexed',...)

【例 4-56】 实现图像的显示块操作

```
I = imread('alumgrns.tif');
I2 = blkproc(I,[8 8],'std2(x)*ones(size(x))');
imshow(I)
figure, imshow(I2,[]);
```

操作结果如图 4-71 所示。

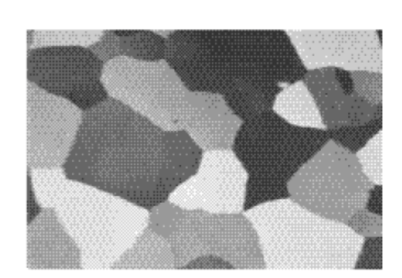
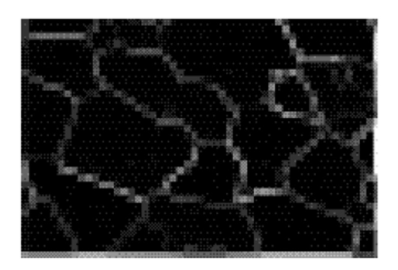



图 4-71 图像的显示块操作

相关命令：

colfilt, nlfilter,inline

4．brighten

功能：增加或降低颜色映像表的亮度。

语法：

brighten(beta)

newmap = brighten(beta)

newmap = brighten(map,beta)

brighten(fig,beta)

相关命令：

imadjust, rgbplot

5．bwarea

功能：计算二进制图像对象的面积。

语法：

total = bwarea(BW)

【例 4-57】 计算二进制图像的面积。

```
BW = imread('circles.tif');
imshow(BW);
```

计算二进制图像的面积如图 4-72 所示。

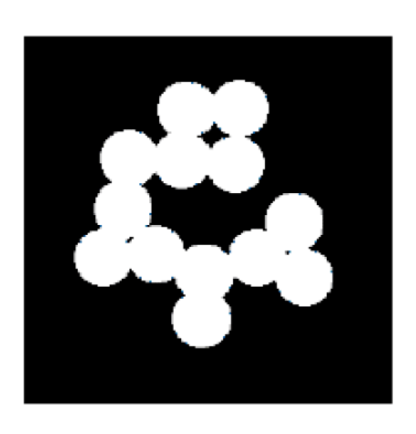


图 4-72　二进制图像的面积

```
bwarea(BW)
ans =
     15799
```

相关命令：

bweuler, bwperim

6．bweuler

功能：计算二进制图像的欧拉数。

语法：

eul = bweuler(BW,n)

【例 4-58】 计算二进制图像的欧拉数。

```
BW = imread('circles.tif');
imshow(BW);
bweuler(BW)
ans =
     -2
```

相关命令：

bwmorph, bwperim

7. bwfill

功能：填充二进制图像的背景色。

语法：

BW2 = bwfill(BW1,c,r,n)

BW2 = bwfill(BW1,n)

[BW2,idx] = bwfill(...)

BW2 = bwfill(x,y,BW1,xi,yi,n)

[x,y,BW2,idx,xi,yi] = bwfill(...)

BW2 = bwfill(BW1,'holes',n)

[BW2,idx] = bwfill(BW1,'holes',n)

【例 4-59】 填充二进制图像的背景色。

```
BW1 =[1 0 0 0 0 0 0 0
1 1 1 1 1 0 0 0
1 0 0 0 1 0 1 0
1 0 0 0 1 1 1 0
1 1 1 1 0 1 1 1
1 0 0 1 1 0 1 0
1 0 0 0 1 0 1 0
1 0 0 0 1 1 1 0]
BW2 = bwfill(BW1,3,3,8)
BW2 =
      1 0 0 0 0 0 0 0
      1 1 1 1 1 0 0 0
      1 1 1 1 1 0 1 0
      1 1 1 1 1 1 1 0
      1 1 1 1 0 1 1 1
      1 0 0 1 1 0 1 0
      1 0 0 0 1 0 1 0
      1 0 0 0 1 1 1 0
I = imread('blood1.tif');
BW3 = ~im2bw(I);
BW4 = bwfill(BW3,'holes');
imshow(BW3)
figure, imshow(BW4)
```

填充结果如图 4-73 所示。

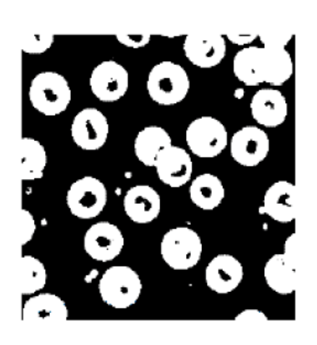
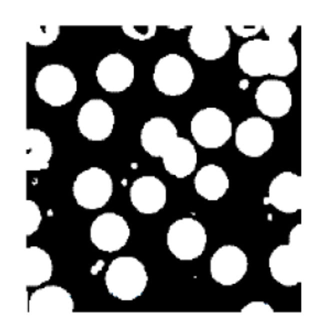



图 4-73　填充二进制图像的背景色

相关命令：

bwselect, roifill

8．bwlabel

功能：标注二进制图像中已连接的部分。

语法：

L = bwlabel(BW,n)

[L,num] = bwlabel(BW,n)

【例 4-60】 标出二进制图像中已连接的部分。

```
BW = [1 1 1 0 0 0 0 0
      1 1 1 0 1 1 0 0
      1 1 1 0 1 1 0 0
      1 1 1 0 0 0 1 0
      1 1 1 0 0 0 1 0
      1 1 1 0 0 0 1 0
      1 1 1 0 0 1 1 0
      1 1 1 0 0 0 0 0]
L = bwlabel(BW,4)
L =
      1 1 1 0 0 0 0 0
      1 1 1 0 2 2 0 0
      1 1 1 0 2 2 0 0
      1 1 1 0 0 0 3 0
      1 1 1 0 0 0 3 0
      1 1 1 0 0 0 3 0
      1 1 1 0 0 3 3 0
      1 1 1 0 0 0 0 0
[r,c] = find(L==2);
rc = [r c]
rc =
      2 5
      3 5
      2 6
      3 6
```

相关命令：

bweuler, bwselect

9．bwmorph

功能：提取二进制图像的轮廓或者是对二值图像进行数学形态学（Mathematical Morphology）运算。

语法格式：

BW1 = bwmorph(BW,operation)

对二值图像进行指定的形态学处理。

BW2 = bwmorph(BW,operation,n)

对二值图像进行 n 次指定的形态学处理，n 可以是 Inf，这种情况下该操作被重复执行直到图像不再发生变化为止。

【例 4-61】 提取二进制图像的轮廓

```
BW1 = imread('circles.tif');
imshow(BW1);
BW2 = bwmorph(BW1,’remove’)
% 'remove'：如果一个像素点的 4 邻域都为 1， 则该像素点将被置 0
BW3 = bwmorph(BW1,’skel’,Inf)
% 'skel'：在这里 n = Inf，提取但保持图像中物体不发生断裂
imshow(BW2)
figure,imshow(BW3)
```

【例 4-62】 二进制图像轮廓的提取

```
% get skeleton of an object in a binary image
close all; clear; clc;
warning off all;
imgdat = logical([1, 0, 0; 1, 0, 1; 0, 0, 1]);
retdat = bwmorph(imgdat, 'bridge') % using 'brige' option
imgdat = logical([0, 0, 0; 0, 1, 0; 0, 0, 0]);
retdat = bwmorph(imgdat, 'clean') % using 'clean' option
imgdat = logical([1, 1, 1; 1, 0, 1; 1, 1, 1]);
retdat = bwmorph(imgdat, 'fill') % using 'fill' option
输出结果：
retdat =
1 1 0
1 1 1
0 1 1
retdat =
0 0 0
0 0 0
0 0 0
retdat =
1 1 1
1 1 1
1 1 1
```

提取结果如图 4-74 所示。





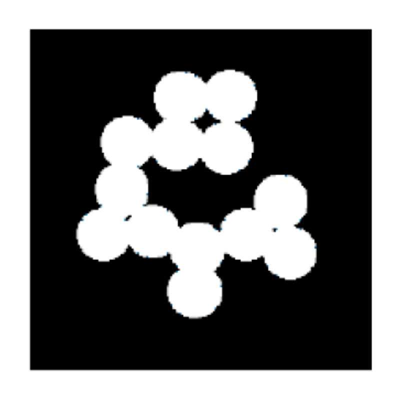


图 4-74　二进制图像轮廓的提取

```
BW2 = bwmorph(BW1,'remove');
BW3 = bwmorph(BW1,'skel',Inf);
imshow(BW2)
figure, imshow(BW3)
```

相关命令：

```
bweuler, bwperim, dilate, erode
```

10. bwperim

功能：计算二进制图像中对象的周长。

语法：

BW2 = bwperim(BW1,n)

【例 4-63】 计算二进制图像中对象的周长。

```
BW1 = imread('circbw.tif');
BW2 = bwperim(BW1,8);
imshow(BW1)
figure, imshow(BW2)
```

计算结果如图 4-75 所示。

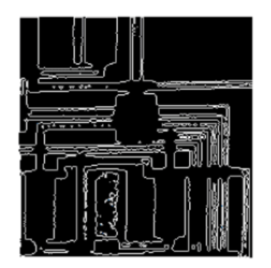



图 4-75　二进制图像中对象的周长

相关命令：

bwarea, bweuler, bwfill

11. bwselect

功能：在二进制图像中选择对象。

语法：

BW2 = bwselect(BW1,c,r,n)

BW2 = bwselect(BW1,n)

[BW2,idx] = bwselect(...)

【例 4-64】 在二进制图像中选择对象。

```
BW1 = imread('text.tif');
c = [16 90 144];
r = [85 197 247];
BW2 = bwselect(BW1,c,r,4);
imshow(BW1)
figure, imshow(BW2)
```

选择对象结果如图 4-76 所示。



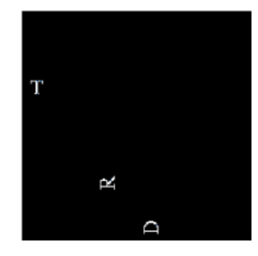


图 4-76 二进制图像中选择对象

相关命令：

bwfill, bwlabel, impixel, roipoly, roifill

12．cmpermute

功能：调整颜色映像表中的颜色。

语法：

[Y,newmap] = cmpermute(X,map)

[Y,newmap] = cmpermute(X,map,index)

【例 4-65】 调整颜色映像表中的颜色。

```
To order a colormap by luminance, use:
ntsc = rgb2ntsc(map);
[dum,index] = sort(ntsc(:,1));
[Y,newmap] = cmpermute(X,map,index);
```

相关命令：

randperm

13．cmunique

功能：查找颜色映像表中特定的颜色及相应的图像。

语法：

[Y,newmap] = cmunique(X,map)

[Y,newmap] = cmunique(RGB)

[Y,newmap] = cmunique(I)

相关命令：

gray2ind, rgb2ind

14．col2im

功能：将矩阵的列重新组织到块中。

语法：

A = col2im(B,[m n],[mm nn],block_type)

A = col2im(B,[m n],[mm nn])

相关命令：

blkproc, colfilt, im2col, nlfilter

15．colfilt

功能：利用列相关函数进行边沿操作。

语法：

B = colfilt(A,[m n],block_type,fun)

B = colfilt(A,[m n],block_type,fun,P1,P2,...)

B = colfilt(A,[m n],[mblock nblock],block_type,fun,...)

B=colfilt(A,'indexed',...)

相关命令：

blkproc,col2im,im2col,nlfilter

16．colorbar

功能：显示颜色条。

语法：

colorbar('vert')，colorbar('horiz')，colorbar(h)，colorbar

h = colorbar(...)，如 I = imread('blood1.tif');

h = fspecial('log')

I2 = filter2(h,I)

imshow(I2,[]), colormap(jet(64)), colorbar

显示颜色条结果如图 4-77 所示。

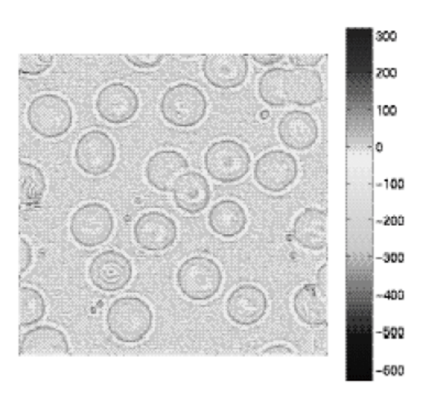


图 4-77　显示颜色条

17．conv2

功能：进行二维卷积操作。

语法：

C = conv2(A,B)

C = conv2(hcol,hrow,A)

C = conv2(...,shape)

【例 4-66】 二维卷积操作。

```
A = magic(5)
A =
```

```
    17 24 1 8 15
    23 5 7 14 16
    4 6 13 20 22
    10 12 19 21 3
    11 18 25 2 9
B = [1 2 1;0 2 0;3 1 3]
B =
    1 2 1
    0 2 0
    3 1 3
C = conv2(A,B)
C =
    17 58 6634 32 38 15
    23 85 88 35 67 76 16
    55 149 117 163 159 135 67
    79 78 160 161 187 129 51
    23 82 153 199 205 108 75
    30 68 135 168 91 84 9
    33 65 126 85 104 15 27
```

相关命令：

filter2

18．convmtx2

功能：计算二维卷积矩阵。

语法：

T = convmtx2(H,m,n)

T = convmtx2(H,[m n])

相关命令：

conv2

19．convn

功能：计算 n 维卷积。

语法：

C = convn(A,B)

C = convn(A,B,shape)

相关命令：

conv2

20．corr2

功能：计算两个矩阵的二维相关系数。

语法：

r = corr2(A,B)

相关命令：

std2

21．dct2

功能：进行二维离散余弦变换。

语法：



B = dct2(A)

B = dct2(A,m,n)

B = dct2(A,[m n])

【例 4-67】 二维离散余弦变换。

```
RGB = imread('autumn.tif');
I = rgb2gray(RGB);
J = dct2(I);
imshow(log(abs(J)),[]), colormap(jet(64)), colorbar
J(abs(J) < 10) = 0;
K = idct2(J)/255;
imshow(K)
```

二维离散余弦变换如图 4-78 所示。变换结果如图 4-79 所示。



图 4-78　二维离散余弦变换

图 4-79　二维离散余弦变换结果

相关命令：

fft2, idct2, ifft2

22. dctmtx

功能：计算离散余弦变换矩阵。

语法：

D = dctmtx(n)

相关命令：

dct2

23. dilate

功能：放大二进制图像。

语法：

BW2 = dilate(BW1,SE)

BW2 = dilate(BW1,SE,alg)

BW2 = dilate(BW1,SE,...,n)

【例 4-68】 放大二进制图像。

```
BW1 = imread('text.tif');
SE = ones(6,2);
```

```
BW2 = dilate(BW1,SE);
imshow(BW1)
figure, imshow(BW2)
```

结果如图 4-80 所示。





图 4-80　二进制图像的放大

相关命令：

bwmorph, erode

24．dither

功能：通过抖动增加外观颜色分辨率，转换图像。

语法：

X = dither(RGB,map)

BW = dither(I)

相关命令：

rgb2ind

25．double

功能：转换数据为双精度型。

语法：

B = double(A)

举例：

A = imread('saturn.tif');

B = sqrt(double(A));

相关命令：

im2double, im2uint, uint8

26．edge

功能：识别强度图像中的边界。

语法：

BW = edge(I,'sobel')

BW = edge(I,'sobel',thresh)

BW = edge(I,'sobel',thresh,direction)

[BW,thresh] = edge(I,'sobel',...)

BW = edge(I,'prewitt')

BW = edge(I,'prewitt',thresh)

BW = edge(I,'prewitt',thresh,direction)

[BW,thresh] = edge(I,'prewitt',...)

BW = edge(I,'roberts')

BW = edge(I,'roberts',thresh)

[BW,thresh] = edge(I,'roberts',...)

BW = edge(I,'log')

BW = edge(I,'log',thresh)

BW = edge(I,'log',thresh,sigma)

[BW,threshold] = edge(I,'log',...)

BW = edge(I,'zerocross',thresh,h)

[BW,thresh] = edge(I,'zerocross',...)

BW = edge(I,'canny')

BW = edge(I,'canny',thresh)

BW = edge(I,'canny',thresh,sigma)

[BW,threshold] = edge(I,'canny',...)

【例 4-69】 识别强度图像中的边界。

```
I = imread('rice.tif');
BW1 = edge(I,'prewitt');
BW2 = edge(I,'canny');
imshow(BW1);
figure, imshow(BW2)
```

识别结果如图 4-81 所示。





图 4-81　图像边界的识别

27．erode

功能：弱化二进制图像的边界。

语法：

BW2 = erode(BW1,SE)

BW2 = erode(BW1,SE,alg)

BW2 = erode(BW1,SE,...,n)

【例 4-70】弱化二进制图像的边界。

```
BW1 = imread('text.tif');
SE = ones(3,1);
BW2 = erode(BW1,SE);
imshow(BW1)
figure, imshow(BW2)
```

弱化结果如图 4-82 所示。





图 4-82　二进制图像边界的弱化

相关命令：

bwmorph, dilate

28. fft2

功能：进行二维快速傅里叶变换。

语法：

B = fft2(A)

B = fft2(A,m,n)

【例 4-71】 二维快速傅里叶变换。

```
load imdemos saturn2
imshow(saturn2)
```

快速傅里叶变换如图 4-83 所示。

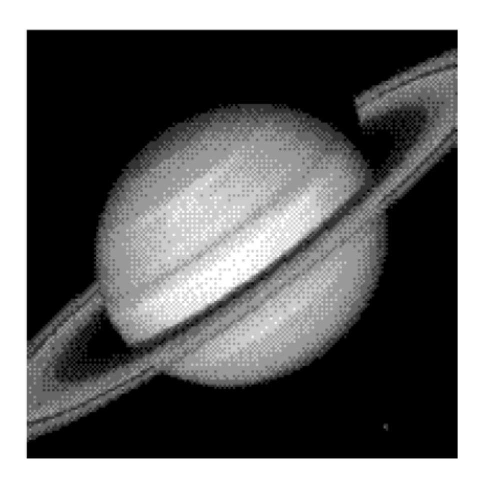


图 4-83　快速傅里叶变换

```
B = fftshift(fft2(saturn2));
imshow(log(abs(B)),[]), colormap(jet(64)), colorbar
```

二维快速傅里叶变换如图 4-84 所示。

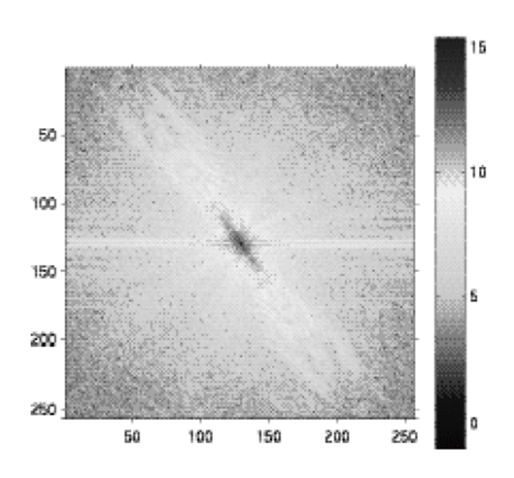


图 4-84　二维快速傅里叶变换

相关命令：

dct2, fftshift, idct2, ifft2

29．fftn

功能：进行 n 维快速傅里叶变换。

语法：

B = fftn(A)

B = fftn(A,siz)

相关命令：

fft2, ifftn

30．fftshift

功能：把快速傅里叶变换的 DC 组件移到光谱中心。

语法：

B = fftshift(A)

【例 4-72】 把快速傅里叶变换的 DC 组件移到光谱中心。

```
B = fftn(A);
C = fftshift(B);
```

相关命令：

fft2, fftn, ifftshift

31．filter2

功能：进行二维线性过滤操作。

语法：

B = filter2(h,A)

B = filter2(h,A,shape)

【例 4-73】 二维线性过滤操作。

```
A = magic(6)
A =
    35  1  6  26  19  24
    3  32  7  21  23  25
    31  9  2  22  27  20
    8  28  33  17  10  15
    30  5  34  12  14  16
    4  36  29  13  18  11
h = fspecial('sobel')
h =
    1  2  1
    0  0  0
    -1  -2  -1
B = filter2(h,A,'valid')
B =
    -8  4  4  -8
    -23  -44  -5  40
    -23  -50  1  40
    -8  4  4  -8
```

相关命令：

conv2, roifilt2

32. freqspace

功能：确定二维频率响应的频率空间。

语法：

[f1,f2] = freqspace(n)

[f1,f2] = freqspace([m n])

[x1,y1] = freqspace(...,'meshgrid')

f = freqspace(N)

f = freqspace(N,'whole')

相关命令：

fsamp2, fwind1, fwind2

33. freqz2

功能：计算二维频率响应。

语法：

[H,f1,f2] = freqz2(h,n1,n2)

[H,f1,f2] = freqz2(h,[n2 n1])

[H,f1,f2] = freqz2(h,f1,f2)

[H,f1,f2] = freqz2(h)

[...] = freqz2(h,...,[dx dy])

[...] = freqz2(h,...,dx)

freqz2(...)

【例 4-74】 计算二维频率响应。

```
Hd = zeros(16,16);
Hd(5:12,5:12) = 1;
Hd(7:10,7:10) = 0;
h = fwind1(Hd,bartlett(16));
colormap(jet(64))
freqz2(h,[32 32]); axis ([-1 1 -1 1 0 1])
```

响应结果如图 4-85 所示。

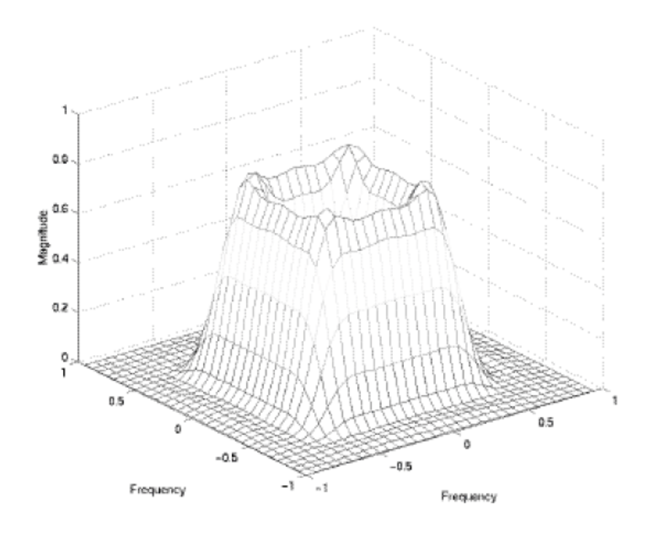


图 4-85　二维频率响应的计算

34．fsamp2

功能：用频率采样法设计二维 FIR 过滤器。

语法：

h = fsamp2(Hd)

h = fsamp2(f1,f2,Hd,[m n])

【例 4-75】 用频率采样法设计二维 FIR 过滤器。

```
[f1,f2] = freqspace(21,'meshgrid');
Hd = ones(21);
r = sqrt(f1.^2 + f2.^2);
Hd((r<0.1)|(r>0.5)) = 0;
colormap(jet(64))
mesh(f1,f2,Hd)
```

相关命令：

conv2, filter2, freqspace, ftrans2, fwind1, fwind2

35．fspecial

功能：创建预定义过滤器。

语法：

h = fspecial(type)

h = fspecial(type,parameters)

【例 4-76】 创建预定义过滤器。

```
I = imread('saturn.tif');
h = fspecial('unsharp',0.5);
I2 = filter2(h,I)/255;
imshow(I)
figure, imshow(I2)
```

结果如图 4-86 所示。

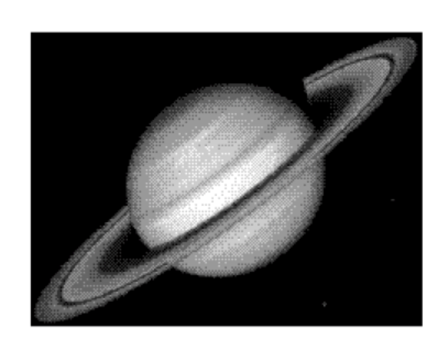
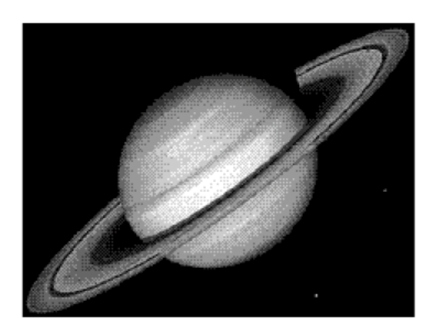



图 4-86 预定义过滤器

相关命令：

conv2, edge, filter2, fsamp2, fwind1, fwind2

36. ftrans2

功能：通过频率转换设计二维 FIR 过滤器。

语法：

h = ftrans2(b,t)

h = ftrans2(b)

【例 4-77】 通过频率转换设计二维 FIR 过滤器。

```
colormap(jet(64))
b = remez(10,[0 0.05 0.15 0.55 0.65 1],[0 0 1 1 0 0]);
[H,w] = freqz(b,1,128,'whole');
plot(w/pi-1,fftshift(abs(H)))
```

相关命令：

conv2, filter2, fsamp2, fwind1, fwind2

37. fwind1

功能：用一维窗口方法设计二维 FIR 过滤器。

语法：

h = fwind1(Hd,win)

h = fwind1(Hd,win1,win2)

h = fwind1(f1,f2,Hd,...)

【例 4-78】 用一维窗口方法设计二维 FIR 过滤器。

```
[f1,f2] = freqspace(21,'meshgrid');
Hd = ones(21);
r = sqrt(f1.^2 + f2.^2);
Hd((r<0.1)|(r>0.5)) = 0;
```

```
colormap(jet(64))
mesh(f1,f2,Hd)
```

相关命令：

conv2, filter2, fsamp2, freqspace, ftrans2, fwind2

38．fwind2

功能：用二维窗口方法设计二维 FIR 过滤器。

语法：

h = fwind2(Hd,win)

h = fwind2(f1,f2,Hd,win)

【例 4-79】 用二维窗口方法设计二维 FIR 过滤器。

```
[f1,f2] = freqspace(21,'meshgrid');
Hd = ones(21);
r = sqrt(f1.^2 + f2.^2);
Hd((r<0.1)|(r>0.5)) = 0;
colormap(jet(64))
mesh(f1,f2,Hd)
```

相关命令：

conv2, filter2, fsamp2, freqspace, ftrans2, fwind1

39．getimage

功能：从坐标轴取得图像数据。

语法：

A = getimage(h)

[x,y,A] = getimage(h)

[...,A,flag] = getimage(h)

[...] = getimage

【例 4-80】 从坐标轴取得图像数据。

```
imshow rice.tif
I = getimage;
```

40．gray2ind

功能：转换灰度图像为索引图像。

语法：

[X,map] = gray2ind(I,n)

相关命令：

ind2gray

41．grayslice

功能：从灰度图像创建索引图像。

语法：

X = grayslice(I,n)

X = grayslice(I,v)

【例 4-81】 从灰度图像创建索引图像。

```
I = imread('ngc4024m.tif');
X = grayslice(I,16);
imshow(I)
figure, imshow(X,jet(16))
```

相关命令：

gray2ind

42．histeq

功能：用柱状图均等化增强对比。

语法：

J = histeq(I,hgram)

J = histeq(I,n)

[J,T] = histeq(I,...)

【例 4-82】 用柱状图均等化增强对比。

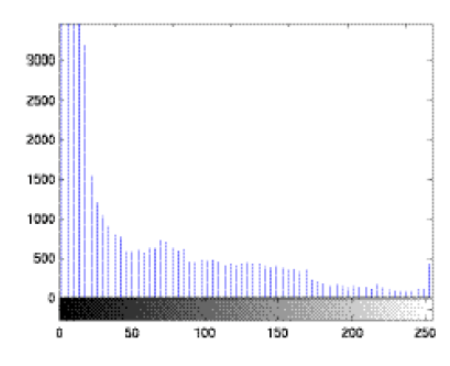
```
I = imread('tire.tif');
J = histeq(I);
imshow(I)
figure, imshow(J)
```

对比结果如图 4-87、图 4-88 所示。




图 4-87　用柱状图均等化增强对比

```
imhist(I,64)
figure; imhist(J,64)
```



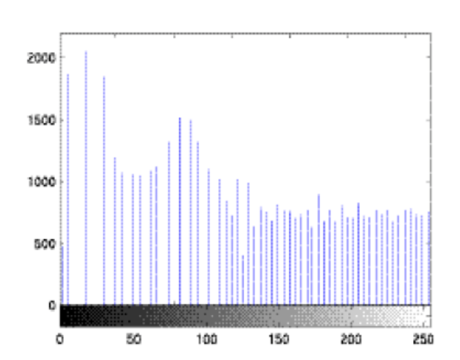


图 4-88　用柱状图均等化增强对比

相关命令：

brighten, imadjust, imhist

43. hsv2rgb

功能：转换 HSV 值为 RGB 颜色空间。

语法：

rgbmap = hsv2rgb(hsvmap)

RGB = hsv2rgb(HSV)

相关命令：

rgb2hsv, rgbplot

44. idct2

功能：计算二维离散反余弦变换。

语法：

B = idct2(A)

B = idct2(A,m,n)

B = idct2(A,[m n])

相关命令：

dct2, dctmtx, fft2, ifft2

45. ifft2

功能：计算二维快速傅里叶反变换。

语法：

B = ifft2(A)

B = ifft2(A,m,n)

相关命令：

fft2, fftshift, idct2

46. ifftn

功能：计算 n 维快速傅里叶反变换。

语法：

B = ifftn(A)

B = ifftn(A,siz)

相关命令：

fft2, fftn, ifft2

47. sim2bw

功能：转换图像为二进制图像。

语法：

BW = im2bw(I,level)

BW = im2bw(X,map,level)

BW = im2bw(RGB,level)

【例 4-83】 转换图像为二进制图像。

```
load trees
BW = im2bw(X,map,0.4);
imshow(X,map)
figure, imshow(BW)
```

转换结果如图 4-89 所示。




图 4-89 转换图像为二进制图像

相关命令：

ind2gray, rgb2gray

48. im2col

功能：重调图像块为列。

语法：

B = im2col(A,[m n],block_type)

B = im2col(A,[m n])

B = im2col(A,'indexed',...)

相关命令：

blkproc, col2im, colfilt, nlfilter

49. im2double

功能：转换图像矩阵为双精度型。

语法：

I2 = im2double(I1)

RGB2 = im2double(RGB1)

BW2 = im2double(BW1)

X2 = im2double(X1,'indexed')

相关命令：

double, im2uint8, uint8

50. im2uint8

功能：转换图像阵列为 8 位无符号整型。

语法：

I2 = im2uint8(I1)

RGB2 = im2uint8(RGB1)

BW2 = im2uint8(BW1)

X2 = im2uint8(X1,'indexed')



相关命令：

im2uint16, double, im2double, uint8, imapprox, uint16

51. im2uint16

功能：转换图像阵列为 16 位无符号整型。

语法：

I2 = im2uint16(I1)

RGB2 = im2uint16(RGB1)

X2 = im2uint16(X1,'indexed')

相关命令：

im2uint8, double, im2double, uint8, uint16, imapprox

52. imadjust

功能：调整图像灰度值或颜色映像表。

语法：

J = imadjust(I,[low high],[bottom top],gamma)

newmap = imadjust(map,[low high],[bottom top],gamma)

RGB2 = imadjust(RGB1,...)

【例 4-84】 调整图像灰度值或颜色映像表。

```
I = imread('pout.tif');
J = imadjust(I,[0.3 0.7],[]);
imshow(I)
figure, imshow(J)
```

调整结果如图 4-90 所示。

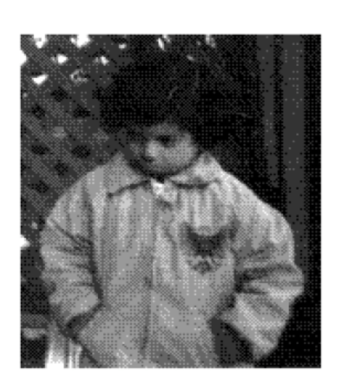



图 4-90　调整图像灰度值

相关命令：

brighten, histeq

53. imapprox

功能：对索引图像进行近似处理。

语法：

[Y,newmap] = imapprox(X,map,n)

[Y,newmap] = imapprox(X,map,tol)

Y = imapprox(X,map,newmap)

[...] = imapprox(...,dither_option)

相关命令：

cmunique, dither, rgb2ind

54．imcontour

功能：创建图像数据的轮廓图。

语法：

imcontour(I,n)

imcontour(I,v)

imcontour(x,y,...)

imcontour(...,LineSpec)

[C,h] = imcontour(...)

【例 4-85】 创建图像数据的轮廓图。

```
I = imread('ic.tif');
imcontour(I,3)
```

结果如图 4-91 所示。

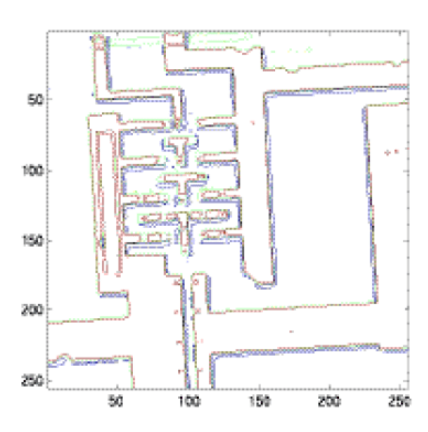


图 4-91　图像数据的轮廓图

相关命令：

clabel, contour, LineSpec

55．imcrop

功能：剪切图像。

语法：

I2 = imcrop(I)

X2 = imcrop(X,map)

RGB2 = imcrop(RGB)

I2 = imcrop(I,rect)

X2 = imcrop(X,map,rect)

RGB2 = imcrop(RGB,rect)

[...] = imcrop(x,y,...)

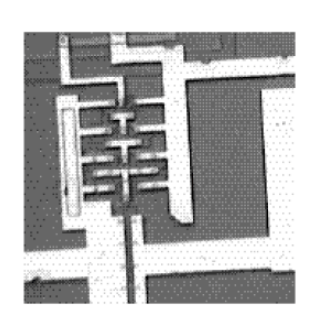
[A,rect] = imcrop(...)

[x,y,A,rect] = imcrop(...)

【例 4-86】 剪切图像。

```
I = imread('ic.tif');
I2 = imcrop(I,[60 40 100 90]);
imshow(I)
figure, imshow(I2)
```

剪切结果如图 4-92 所示。



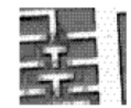


图 4-92　图像的剪切

相关命令：

zoom

56．imfeature

功能：计算图像区域的特征尺寸。

语法：

stats = imfeature(L,measurements)

stats = imfeature(L,measurements,n)

【例 4-87】 计算图像区域的特征尺寸。

```
BW = imread('text.tif');
L = bwlabel(BW);
stats = imfeature(L,'all');
stats(23)
ans =
Area: 89
Centroid: [95.6742 192.9775]
BoundingBox: [87.5000 184.5000 16 15]
MajorAxisLength: 19.9127
MinorAxisLength: 14.2953
Eccentricity: 0.6961
Orientation: 9.0845
ConvexHull: [28x2 double]
ConvexImage: [15x16 uint8 ]
ConvexArea: 205
Image: [15x16 uint8 ]
FilledImage: [15x16 uint8 ]
FilledArea: 122
```

```
EulerNumber: 0
Extrema: [ 8x2 double]
EquivDiameter: 10.6451
Solidity: 0.4341
Extent: 0.3708
PixelList: [89x2 double]
```

相关命令：

bwlabel

57．imfinfo

功能：返回图形文件信息。

语法：

info = imfinfo(filename,fmt)

info = imfinfo(filename)

【例 4-88】 返回图形文件信息。

```
info = imfinfo('canoe.tif')
info =
Filename:'canoe.tif'
FileModDate: '25-Oct-1996 22:10:39'
FileSize: 69708
Format: 'tif'
FormatVersion: []
Width: 346
Height: 207
BitDepth: 8
ColorType: 'indexed'
FormatSignature: [73 73 42 0]
ByteOrder: 'little-endian'
NewSubfileType: 0
BitsPerSample: 8
Compression: 'PackBits'
PhotometricInterpretation: 'RGB Palette'
StripOffsets: [ 9x1 double]
SamplesPerPixel: 1
RowsPerStrip: 23
StripByteCounts: [ 9x1 double]
XResolution: 72
YResolution: 72
ResolutionUnit: 'Inch'
Colormap: [256x3 double]
PlanarConfiguration: 'Chunky'
TileWidth: []
TileLength: []
TileOffsets: []
TileByteCounts: []
Orientation: 1
FillOrder: 1
GrayResponseUnit: 0.0100
MaxSampleValue: 255
```

```
MinSampleValue: 0
Thresholding: 1
```

相关命令：

imread, imwrite

58．imhist

功能：显示图像数据的柱状图。

语法：

imhist(I,n)

imhist(X,map)

[counts,x] = imhist(...)

【例 4-89】 显示图像数据的柱状图。

```
I = imread('pout.tif');
imhist(I)
```

显示结果如图 4-93 所示。

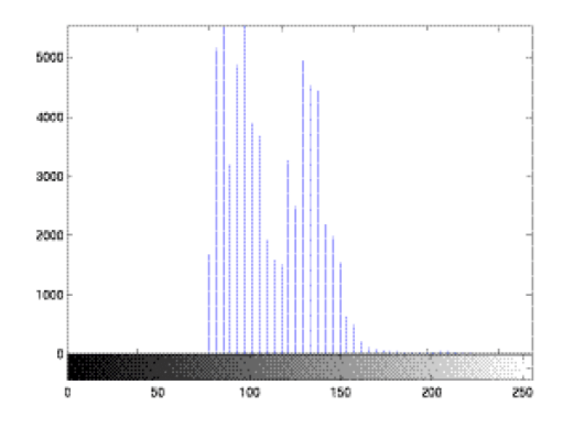


图 4-93　图像数据的柱状图显示

相关命令：

histeq

59．immovie

功能：创建多帧索引图的电影动画。

语法：

mov = immovie(X,map)

【例 4-90】 创建多帧索引图的电影动画。

```
load mri
mov = immovie(D,map);
```

相关命令：

montage

60．imnoise

功能：增加图像的渲染效果。

语法：

J = imnoise(I,type)

J = imnoise(I,type,parameters)

【例 4-91】 增加图像的渲染效果。

```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.02);
imshow(I)
figure, imshow(J)
```

显示结果如图 4-94 所示。

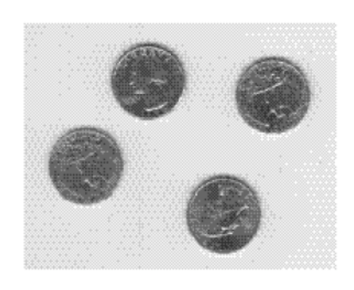
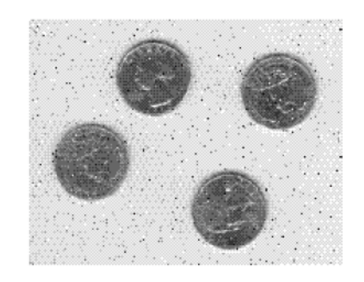



图 4-94　图像的渲染效果

相关命令：

rand

61．impixel

功能：确定像素颜色值。

语法：

P = impixel(I)

P = impixel(X,map)

P = impixel(RGB)

P = impixel(I,c,r)

P = impixel(X,map,c,r)

P = impixel(RGB,c,r)

[c,r,P] = impixel(...)

P = impixel(x,y,I,xi,yi)

P = impixel(x,y,X,map,xi,yi)

P = impixel(x,y,RGB,xi,yi)

[xi,yi,P] = impixel(x,y,...)

【例 4-92】 确定像素颜色值。

```
RGB = imread('flowers.tif');
c = [12 146 410];
r = [104 156 129];
pixels = impixel(RGB,c,r)
pixels =
61 59 101
253 240 0
```

```
237 37 44
```

相关命令：

improfile, pixval

62. improfile

功能：沿线段计算剖面图的像素值。

语法：

c = improfile

c = improfile(n)

c = improfile(I,xi,yi)

c = improfile(I,xi,yi,n)

[cx,cy,c] = improfile(...)

[cx,cy,c,xi,yi] = improfile(...)

[...] = improfile(x,y,I,xi,yi)

[...] = improfile(x,y,I,xi,yi,n)

[...] = improfile(...,method)

【例 4-93】 沿线段计算剖面图的像素值。

```
I = imread('alumgrns.tif');
x = [35 338 346 103];
y = [253 250 17 148];
improfile(I,x,y), grid on
```

显示结果如图 4-95 所示。

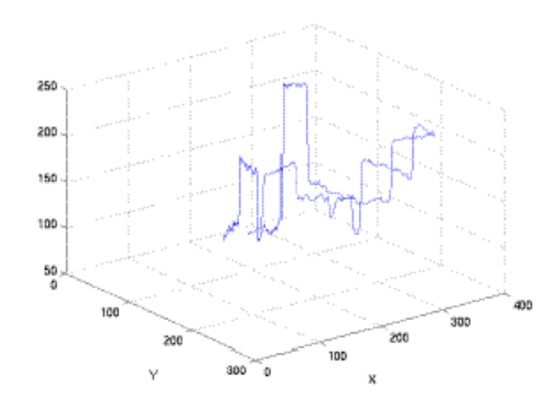


图 4-95　剖面图的像素值计算

相关命令：

impixel, pixval

63. imread

功能：从图形文件中读取图像。

语法：

A = imread(filename,fmt)

[X,map] = imread(filename,fmt)

[...] = imread(filename)

[...] = imread(...,idx) (TIFF only)

[...] = imread(...,ref) (HDF only)

[...] = imread(...,'BackgroundColor',BG) (PNG only)

[A,map,alpha] = imread(...) (PNG only)

【例 4-94】 从图形文件中读取图像。

```
[X,map] = imread('flowers.tif',6);
info = imfinfo('skull.hdf');
[X,map] = imread('skull.hdf',info(4).Reference);
bg = [255 0 0];
A = imread('image.png','BackgroundColor',bg);
[A,map,alpha] = imread('image.png');
```

相关命令：

imfinfo, imwrite,fread,double,uint8,uint16

64. imresize

功能：改变图像大小。

语法：

B = imresize(A,m,method)

B = imresize(A,[mrows ncols],method)

B = imresize(...,method,n)

B = imresize(...,method,h)

65. imrotate

功能：旋转图像。

语法：

B = imrotate(A,angle,method)

B = imrotate(A,angle,method,'crop')

【例 4-95】 旋转图像。

```
I = imread('ic.tif');
J = imrotate(I,-4,'bilinear','crop');
imshow(I)
figure, imshow(J)
```

旋转图像如图 4-96 所示。

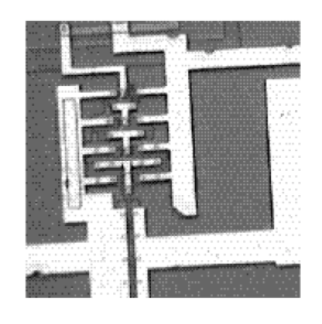
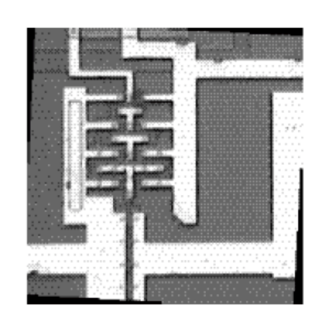



图 4-96　图像旋转

相关命令：

imcrop, imresize

66. imshow

功能：显示图像。

语法：

imshow(I,n)

imshow(I,[low high])

imshow(BW)

imshow(X,map)

imshow(RGB)

imshow(...,display_option)

imshow(x,y,A,...)

imshow filename

h = imshow(...)

相关命令：

getimage, imread, iptgetpref, iptsetpref, subimage, truesize, warp

67. imwrite

功能：把图像写入图形文件中。

语法：

imwrite(A,filename,fmt)

imwrite(X,map,filename,fmt)

imwrite(...,filename)

imwrite(...,Param1,Val1,Param2,Val2...)

【例 4-96】 把图像写入图形文件中。

```
imwrite(X,map,'flowers.hdf','Compression','none',...
'WriteMode','append')
```

相关命令：

imfinfo, imread

68. ind2gray

功能：把检索图像转化为灰度图像。

语法：

I = ind2gray(X,map)

【例 4-97】 把检索图像转化为灰度图像。

```
load trees
I = ind2gray(X,map);
imshow(X,map)
figure,imshow(I)
```

显示结果如图 4-97 所示。




图 4-97　检索图像转化为灰度图像

相关命令：

gray2ind, imshow, rgb2ntsc

69．ind2rgb

功能：转化索引图像为 RGB 真彩图像。

语法：

RGB = ind2rgb(X,map)

相关命令：

ind2gray, rgb2ind

70．iptgetpref

功能：获取图像处理工具箱参数设置。

语法：

value = iptgetpref(prefname)

【例 4-98】 获取图像处理工具箱参数设置。

```
value = iptgetpref('ImshowAxesVisible')
value =
       off
```

相关命令：

imshow, iptsetpref

71．iptsetpref

功能：设置图像处理工具箱参数。

语法：

iptsetpref(prefname,value)

【例 4-99】 设置图像处理工具箱参数。

```
iptsetpref('ImshowBorder','tight')
```

相关命令：

imshow, iptgetpref, truesize

72．iradon

功能：进行反 Radon 变换。

语法：

I = iradon(P,theta)

I = iradon(P,theta,interp,filter,d,n)

[I,h] = iradon(...)

【例 4-100】 进行反 Radon 变换。

```
P = phantom(128);
R = radon(P,0:179);
I = iradon(R,0:179,'nearest','Hann');
imshow(P)
figure, imshow(I)
```

变换对应如图 4-98 所示。

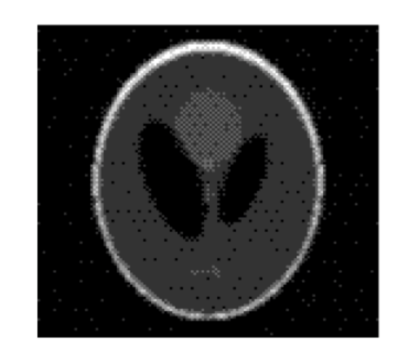



图 4-98　反 Radon 变换

相关命令：

radon, phantom

73. isbw

功能：判断是否为二进制图像。

语法：

flag = isbw(A)

相关命令：

isind, isgray, isrgb

74. isgray

功能：判断是否为灰度图像。

语法：

flag = isgray(A)

相关命令：

isbw, isind, isrgb

75. isind

功能：判断是否为索引图像。

语法：

flag = isind(A)

相关命令：

isbw, isgray, isrgb

76．isrgb

功能：判断是否为 RGB 真彩图像。

语法：

flag = isrgb(A)

相关命令：

isbw, isgray, isind

77．makelut

功能：创建一个用于 applylut 函数的 lookup 表。

语法：

lut = makelut(fun,n)

lut = makelut(fun,n,P1,P2,...)

【例 4-101】 创建一个用于 applylut 函数的 lookup 表。

```
f = inline('sum(x(:)) >= 2');
lut = makelut(f,2)
lut =
      0
      0
      0
      1
      0
      1
      1
      1
      0
      1
      1
      1
      1
      1
      1
      1
```

相关命令：

applylut

78．mat2gray

功能：转化矩阵为灰度图像。

语法：

I = mat2gray(A,[amin amax])

I = mat2gray(A)

【例 4-102】 转化矩阵为灰度图像。

```
I = imread('rice.tif');
J = filter2(fspecial('sobel'),I);
K = mat2gray(J);
imshow(I)
figure, imshow(K)
```



显示结果如图 4-99 所示。

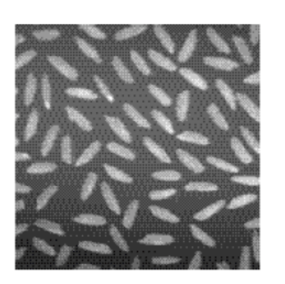
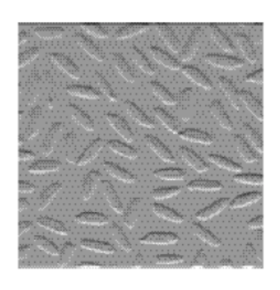



图 4-99　转化矩阵为灰度图像

相关命令：

gray2ind

79．mean2

功能：计算矩阵元素的平均值。

语法：

b = mean2(A)

相关命令：

std2, mean, std

80．medfilt2

功能：进行二维中值过滤。

语法：

B = medfilt2(A,[m n])

B = medfilt2(A)

B = medfilt2(A,'indexed',...)

【例 4-103】 二维中值过滤。

```
I = imread('eight.tif');
J = imnoise(I,'salt & pepper',0.02);
K = medfilt2(J);
imshow(J)
figure, imshow(K)
```

过滤结果如图 4-100 所示。

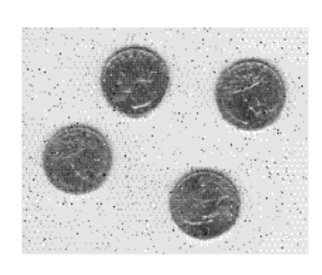
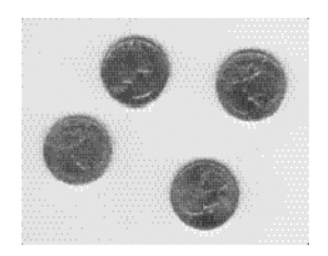



图 4-100　二维中值过滤

相关命令：

filter2, ordfilt2, wiener2

81．montage

功能：在矩形框中同时显示多幅图像。

语法：

montage(I)

montage(BW)

montage(X,map)

montage(RGB)

h = montage(...)

【例 4-104】 在矩形框中同时显示多幅图像。

```
load mri
montage(D,map)
```

显示结果如图 4-101 所示。

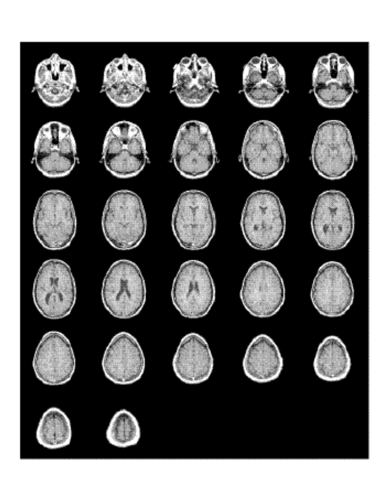


图 4-101　多幅图像显示

相关命令：

immovie

82．nlfilter

功能：进行边沿操作。

语法：

B = nlfilter(A,[m n],fun)

B = nlfilter(A,[m n],fun,P1,P2,...)

B = nlfilter(A,'indexed',...)

【例 4-105】 边沿操作。

```
B = nlfilter(A,[3 3],'median(x(:))');
```

相关命令：

blkproc, colfilt

83．ntsc2rgb

功能：转换 NTSC 的值为 RGB 颜色空间。

语法：

rgbmap = ntsc2rgb(yiqmap)

RGB = ntsc2rgb(YIQ)

相关命令：

rgb2ntsc, rgb2ind, ind2rgb, ind2gray

84．ordfilt2

功能：进行二维统计顺序过滤。

语法：

B = ordfilt2(A,order,domain)

B = ordfilt2(A,order,domain,S)

B = ordfilt2(...,padopt)

相关命令：

medfilt2

85．phantom

功能：产生一个头部幻影图像。

语法：

P = phantom(def,n)

P = phantom(E,n)

[P,E] = phantom(...)

【例 4-106】 产生一个头部幻影图像。

```
P = phantom('Modified Shepp-Logan',200);
```

imshow(P)

显示效果如图 4-102 所示。



图 4-102　头部幻影图像显示

相关命令：

radon, iradon

86．pixval

功能：显示图像像素信息。

语法：

pixval on

pixval off

pixval

pixval(fig,option)

相关命令：

impixel, improfile

87．qtdecomp

功能：进行四叉树分解。

语法：

S = qtdecomp(I)

S = qtdecomp(I,threshold)

S = qtdecomp(I,threshold,mindim)

S = qtdecomp(I,threshold,[mindim maxdim])

S = qtdecomp(I,fun)

S = qtdecomp(I,fun,P1,P2,...)

【例 4-107】 四叉树分解。

```
I = [1  1   1  1   2   3  6   6
     1  1   2  1   4   5  6   8
     1  1   1  1   10 15  7   7
     1  1   1  1   20 25  7   7
     20 22 20 22 1   2  3   4
     20 22 22 20 5   6  7   8
     20 22 20 20 9   10 11 12
     22 22 20 20 13  14 15 16];
S = qtdecomp(I,5);
full(S)
ans =
    4  0  0  0  2  0  2  0
    0  0  0  0  0  0  0  0
    0  0  0  0  1  1  2  0
    0  0  0  0  1  1  0  0
    4  0  0  0  2  0  2  0
    0  0  0  0  0  0  0  0
    0  0  0  0  2  0  2  0
    0  0  0  0  0  0  0  0
```

相关命令：

qtgetblk, qtsetblk

88．qtgetblk

功能：获取四叉树分解中的块值。

语法：

[vals,r,c] = qtgetblk(I,S,dim)

[vals,idx] = qtgetblk(I,S,dim)

【例 4-108】 获取四叉树分解中的块值。

```
[vals,r,c] = qtgetblk(I,S,4)
vals(:,:,1) =
     1  1  1  1
     1  1  2  1
     1  1  1  1
     1  1  1  1
vals(:,:,2) =
     20  22  20  22
     20  22  22  20
     20  22  20  20
     22  22  20  20
r =
     1
     5
c =
     1
     1
```

相关命令：

qtdecomp, qtsetblk

89．qtsetblk

功能：设置四叉树分解中的块值。

语法：

J = qtsetblk(I,S,dim,vals)

【例 4-109】 设置四叉树分解中的块值。

```
newvals = cat(3,zeros(4),ones(4));
J = qtsetblk(I,S,4,newvals)
J =
    0  0  0  0  2   3   6   6
    0  0  0  0  4   5   6   8
    0  0  0  0  10  15  7   7
    0  0  0  0  20  25  7   7
    1  1  1  1  1   2   3   4
    1  1  1  1  5   6   7   8
    1  1  1  1  9   10  11  12
    1  1  1  1  13  14  15  16
```

相关命令：

qtdecomp, qtgetblk

90．radon

功能：计算 Radon 变换。

语法：

R = radon(I,theta)

R = radon(I,theta,n)

[R,xp] = radon(...)

【例 4-110】 计算 Radon 变换。

```
iptsetpref('ImshowAxesVisible','on')
I = zeros(100,100);
I(25:75,25:75) = 1;
theta = 0:180;
[R,xp] = radon(I,theta);
imshow(theta,xp,R,[]), colormap(hot), colorbar
```

变换结果如图 4-103 所示。

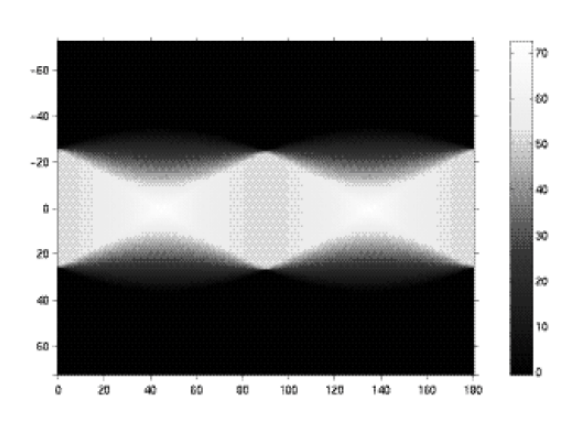


图 4-103　Radon 变换

相关命令：

iradon, phantom

91．rgb2gray

功能：转换 RGB 图像或颜色映像表为灰度图像。

语法：

I = rgb2gray(RGB)

newmap = rgb2gray(map)

相关命令：

ind2gray, ntsc2rgb, rgb2ind, rgb2ntsc

92．rgb2hsv

功能：转化 RGB 值为 HSV 颜色空间。

语法：

hsvmap = rgb2hsv(rgbmap)

HSV = rgb2hsv(RGB)

相关命令：

hsv2rgb, rgbplot

93．rgb2ind

功能：转化 RGB 图像为索引图像。

语法：

[X,map] = rgb2ind(RGB,tol)

[X,map] = rgb2ind(RGB,n)

X = rgb2ind(RGB,map)

[...] = rgb2ind(...,dither_option)

【例 4-111】 转化 RGB 图像为索引图像。

```
RGB = imread('flowers.tif');
[X,map] = rgb2ind(RGB,128);
imshow(X,map)
```

转换结果如图 4-104 所示。



图 4-104　RGB 图像转换为索引图像

相关命令：

cmunique, dither, imapprox, ind2rgb, rgb2gray

94．rgb2ntsc

功能：转化 RGB 的值为 NTSC 颜色空间。

语法：

yiqmap = rgb2ntsc(rgbmap)

YIQ = rgb2ntsc(RGB)

相关命令：

ntsc2rgb, rgb2ind, ind2rgb, ind2gray

95．rgb2ycbcr

功能：转化 RGB 的值为 YCBCR 颜色空间。

语法：

ycbcrmap = rgb2ycbcr(rgbmap)

YCBCR = rgb2ycbcr(RGB)

相关命令：

ntsc2rgb, rgb2ntsc, ycbcr2rgb

96．rgbplot

功能：划分颜色映像表。

语法：

rgbplot(map)

【例 4-112】 划分颜色映像表。

```
rgbplot(jet)
```

划分结果如图 4-105 所示。

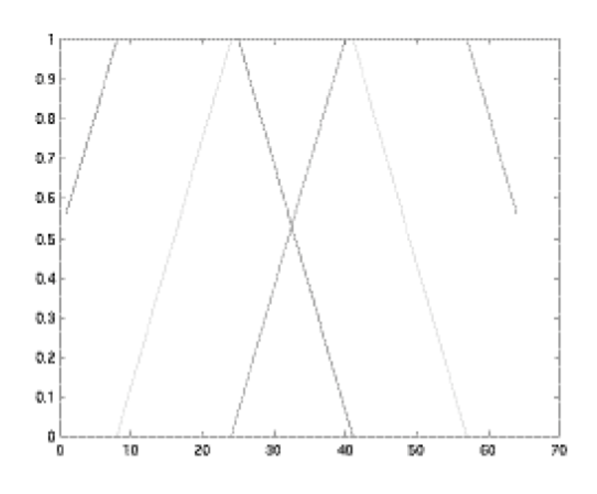


图 4-105　颜色映像表划分

相关命令：

colormap

97. roicolor

功能：选择感兴趣的颜色区。

语法：

BW = roicolor(A,low,high)

BW = roicolor(A,v)

【例 4-113】 选择感兴趣的颜色区。

```
I = imread('rice.tif');
BW = roicolor(I,128,255);
imshow(I);
figure, imshow(BW)
```

显示结果如图 4-106 所示。

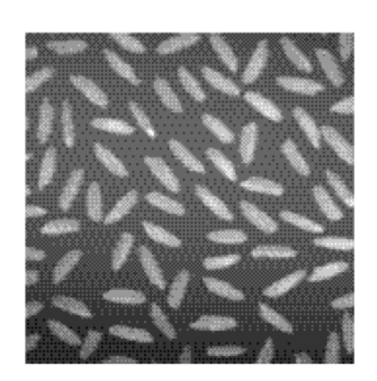




图 4-106　颜色区的选择

相关命令：

roifilt2, roipoly

98．roifill

功能：在图像的任意区域中进行平滑插补。

语法：

J = roifill(I,c,r)

J = roifill(I)

J = roifill(I,BW)

[J,BW] = roifill(...)

J = roifill(x,y,I,xi,yi)

[x,y,J,BW,xi,yi] = roifill(...)

【例 4-114】 在图像的任意区域中进行平滑插补。

```
I = imread('eight.tif');
c = [222 272 300 270 221 194];
r = [21 21 75 121 121 75];
J = roifill(I,c,r);
imshow(I)
figure, imshow(J)
```

插补结果如图 4-107 所示。

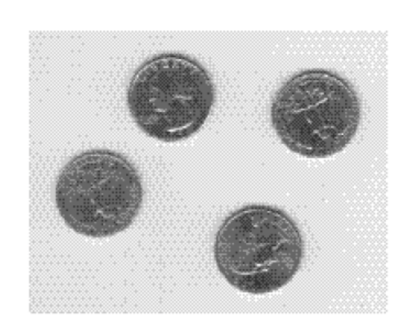
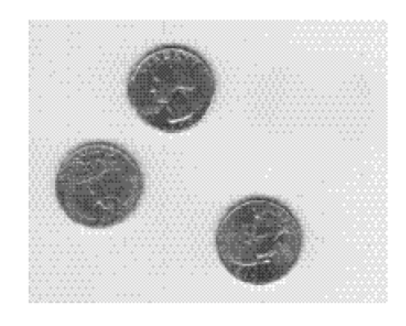



图 4-107　图像区域中的平滑插补

相关命令：

roifilt2, roipoly

99．roifilt2

功能：过滤敏感区域。

语法：

J = roifilt2(h,I,BW)

J = roifilt2(I,BW,fun)

J = roifilt2(I,BW,fun,P1,P2,...)

【例 4-115】 过滤敏感区域。

```
h = fspecial('unsharp');
J = roifilt2(h,I,BW);
imshow(J)
```

过滤结果如图 4-108 所示。

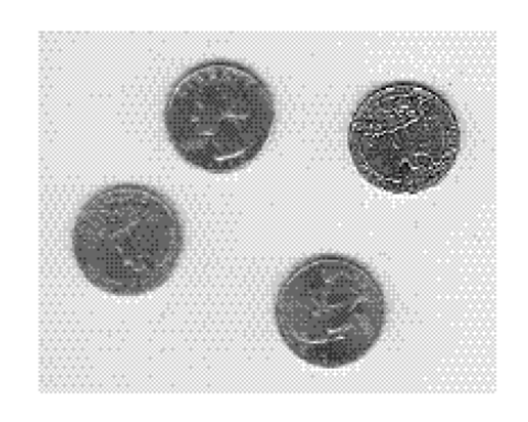


图 4-108　过滤敏感区域

相关命令：

filter2, roipoly

100．roipoly

功能：选择一个敏感的多边形区域。

语法：

BW = roipoly(I,c,r)

BW = roipoly(I)

BW = roipoly(x,y,I,xi,yi)

[BW,xi,yi] = roipoly(...)

[x,y,BW,xi,yi] = roipoly(...)

【例 4-116】 选择一个敏感的多边形区域。

```
I = imread('eight.tif');
c = [222 272 300 270 221 194];
r = [21 21 75 121 121 75];
BW = roipoly(I,c,r);
imshow(I)
figure, imshow(BW)
```

选择结果如图 4-109 所示。

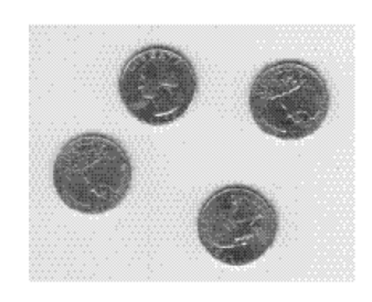



图 4-109　敏感的多边形区域选择

相关命令：

roifilt2, roicolor, roifill

101．std2

功能：计算矩阵元素的标准偏移。

语法：

b = std2(A)

相关命令：

corr2, mean2

102．subimage

功能：在一幅图中显示多个图像。

语法：

subimage(X,map)

subimage(I)

subimage(BW)

subimage(RGB)

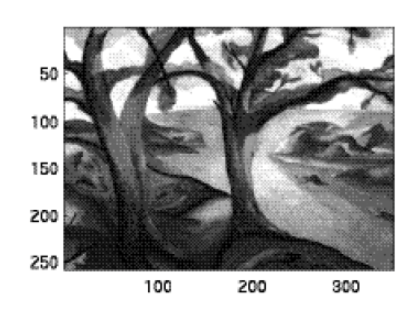
subimage(x,y,...)

h = subimage(...)

【例 4-117】 在一幅图中显示多个图像。

```
load trees
[X2,map2] = imread('forest.tif');
subplot(1,2,1), subimage(X,map)
subplot(1,2,2), subimage(X2,map2)
```

显示结果如图 4-110 所示。



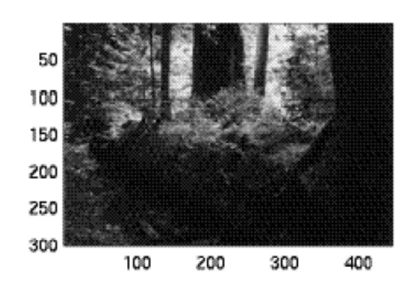


图 4-110　多个图像的累加显示

103．truesize

功能：调整图像显示尺寸。

语法：

truesize(fig,[mrows mcols])

truesize(fig)

相关命令：

imshow, iptsetpref, iptgetpref

104．uint8

功能：转换数据为 8 位无符号整型。

语法：

B = uint8(A)

【例 4-118】 转换数据为 8 位无符号整型。

```
a = [1 3 5];
b = uint8(a);
whos
Name  Size  Bytes  Class
a     1x3   24     doublearray
b     1x3   3      uint8 array
```

相关命令：

double, im2double, im2uint8

105. uint16

功能：转换数据为 16 位无符号整型。

语法：

I = uint16(X)

【例 4-119】 转换数据为 16 位无符号整型。

```
a = [1 3 5];
b = uint16(a);
whos
Name  Size  Bytes  Class
a     1x3   24     double array
b     1x3   6      uint16 array
```

相关命令：

double, datatypes, uint8, uint32, int8, int16, int32.

106. warp

功能：将图像显示到纹理映射表面。

语法：

warp(X,map)

warp(I,n)

warp(BW)

warp(RGB)

warp(z,...)

warp(x,y,z,...)

h = warp(...)

【例 4-120】 将图像显示到纹理映射表面。

```
[x,y,z] = cylinder;
I = imread('testpat1.tif');
```



```
warp(x,y,z,I);
```

结果如图 4-111 所示。

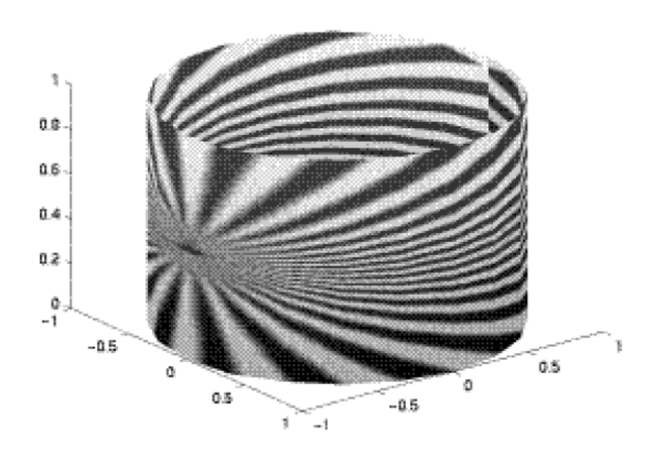


图 4-111　图像显示到纹理映射表面

相关命令：

imshow

107．wiener2

功能：进行二维适应性去噪过滤处理。

语法：

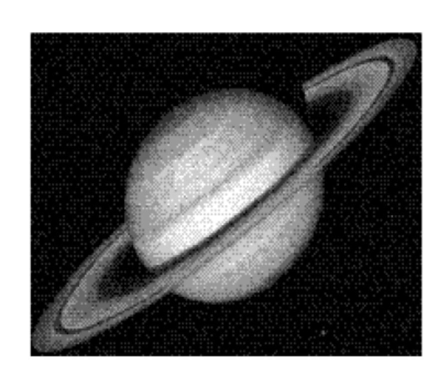
J = wiener2(I,[m n],noise)

[J,noise] = wiener2(I,[m n])

【例 4-121】 进行二维适应性去噪过滤处理。

```
I = imread('saturn.tif');
J = imnoise(I,'gaussian',0,0.005);
K = wiener2(J,[5 5]);
imshow(J)
figure, imshow(K)
```

过滤结果如图 4-112 所示。



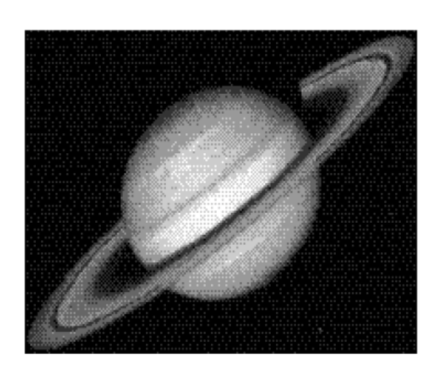


图 4-112　适应性去噪过滤

相关命令：

filter2, medfilt2

108. ycbcr2rgb

功能：转化 YcbCr 值为 RGB 颜色空间。

语法：

```
rgbmap = ycbcr2rgb(ycbcrmap)
RGB = ycbcr2rgb(YCBCR)
```

相关命令：

ntsc2rgb, rgb2ntsc, rgb2ycbcr

109. zoom

功能：缩放图像。

语法：

```
zoom on
zoom off
zoom out
zoom reset
zoom
zoom xon
zoom yon
zoom(factor)
zoom(fig,option)
```

相关命令：

imcrop

4.8 本章小结

通过本章学习 MATLAB，读者初步掌握了有关图像处理与图像分析的基本概念、基础理论和实用技术，了解和掌握图像处理的方法及手段，深刻体会到 MATLAB 是一款基于矩阵数学运算的仿真综合处理软件，图像处理模块可以应用于航空、国防、影像通讯等各个图像处理应用方面。MATLAB 提供的图像处理函数包括排列、变换和锐化等操作，同样利用这些函数能够完成裁减图像和尺寸变换等操作。利用 MATLAB 的设计理念，从矩阵的运算出发，对图像进行处理，其中涵盖内容全面，有助于读者对图像处理技术有一个更加深刻的认识，从本质出发看待问题。

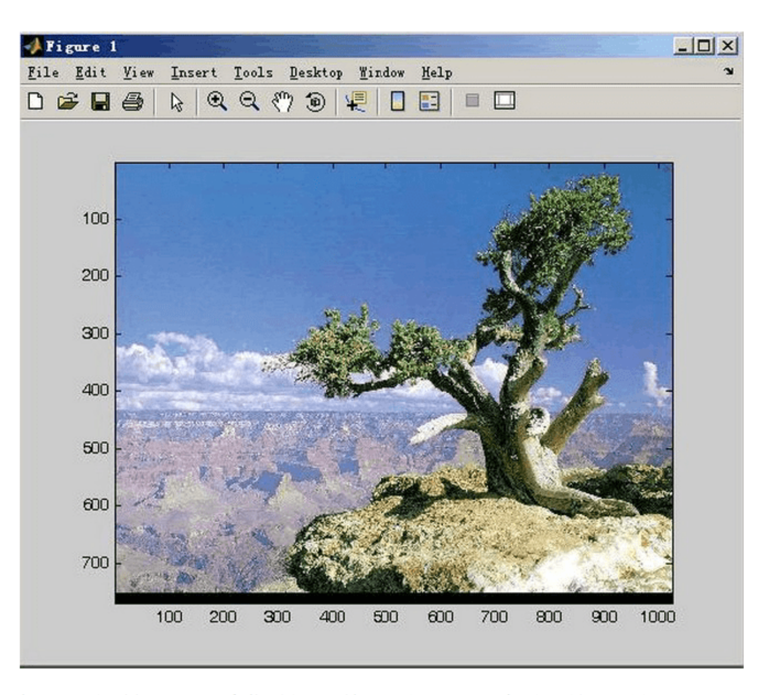
4.9 习题

（1）采集一张格式为*.jpg 的图像，用 MATLAB 的 imread 函数读入图像文件，并用 image 函数显示图像。

解题提示：

```
>> i=imread('eee.jpg');
>> image(i)
```

显示的图像

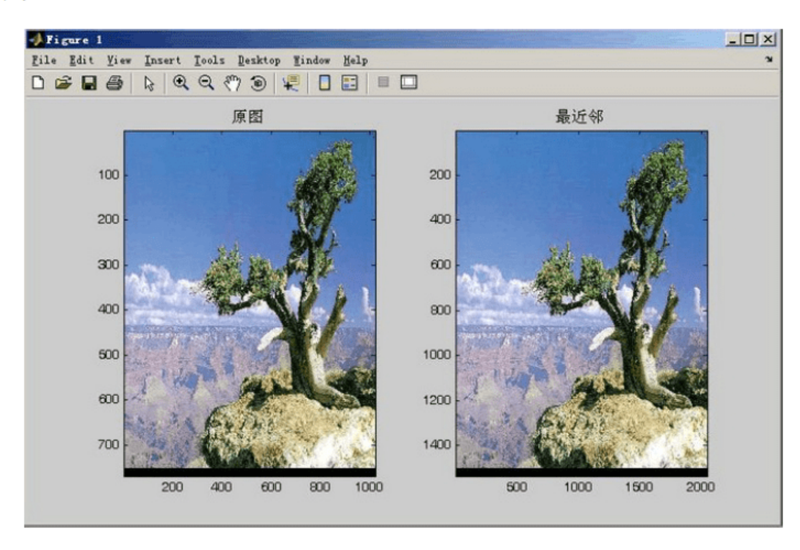


（2）试编写一个 M 文件，对采集的图像进行最近邻插值，并且显示出来与原图像进行对比。

解题提示：

```
>> j=imresize(i,2,'nearest');
>> subplot(1,2,1),image(i),title('原图'), subplot(1,2,2),image(j),
title('最近邻')
```

显示的图像为

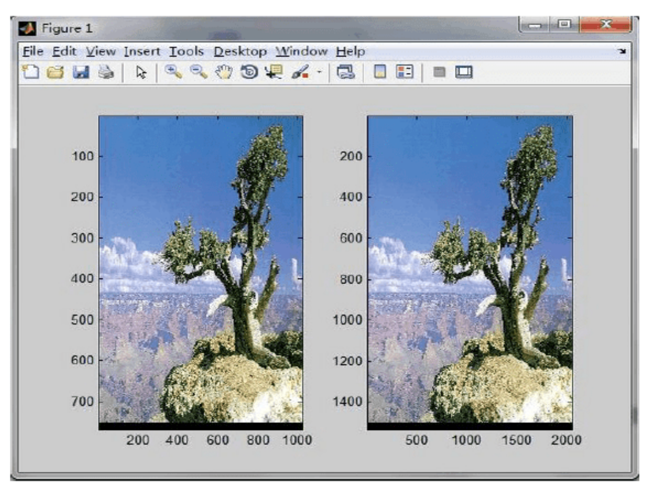


（3）编写程序，对采集的图像进行最近邻插值。

解题提示：

```
>> [r,c]=size('eee.jpg')
```

```
>>  for i=1:r
for j=1:c
              B(i,2*j)=eee(i,j);
B(i,2*j-1)=eee(i,j);
end
end
for i=1:r
 for j=1:2*c
 C(2*i,j)=B(i,j);
C(2*i-1,j)=B(i,j);
end
end
subplot(1,2,1);
>> imshow(eee);
>> subplot(1,2,2);
 >> imshow(C)
```



（4）MATLAB 可以将图像数据进行压缩处理，分析下面的代码说明压缩的原理代码。

```
I = imread('cameraman.tif');
I = im2double(I);
T = dctmtx(8);
B = blkproc(I,[8 8],'P1*x*P2',T,T');
mask =  [
1 1 1 1 0 0 0 0
1 1 1 0 0 0 0 0
1 1 0 0 0 0 0 0
1 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0
];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I2 = blkproc(B2,[8 8],'P1*x*P2',T',T);
imshow(I),
figure, imshow(I2)
```

（5）怎样可以自动获得由鼠标在图像上任意指定的两像素点之间的距离？

第 5 章　M 文件编程

简单地说，M 文件就是用户把要实现的命令写在一个以 .m 作为文件扩展名的文件中，然后由 MATLAB 系统进行解释，运行出结果，实际上 M 文件是一个命令集，因此，MATLAB 具有强大的可开发性与可扩展性。MATLAB 中的许多函数本身都是由 M 文件扩展而成的，而用户也可以利用 M 文件来生成和扩充自己的函数库。

5.1　编程概述

MATLAB 作为一种应用广泛的科学计算软件，不仅可以通过直接交互的指令和操作方式进行强大的数值计算、绘图等，还可以像 C、C++等高级程序语言一样，根据自己的语法规则来进行程序设计。编写的程序文件以 .m 作为扩展名，称之为 M 文件。通过编写 M 文件，用户可以像编写批处理命令一样，将多个 MATLAB 命令集中在一个文件中，既能方便地进行调用，又便于修改；还可以根据用户自身的情况，编写用于解决特定问题的 M 文件，这样就实现了结构化程序设计，并降低代码重用率。实际上，MATLAB 自带的许多函数就是 M 函数文件。MATLAB 提供的编辑器可以使用户方便地进行 M 文件的编写。

5.1.1　M 文件的创建及运行

建立 M 文件的必要性。当遇到输入命令较多及重复输入命令的情况时，利用命令文件就显得很方便了。将所有要执行的命令按顺序放到一个扩展名为.m 的文本文件中，每次运行时只需在 MATLAB 的命令窗口输入 m 文件的文件名就可以了。注意，m 文件最好直接放在 MATLAB 的默认搜索路径下（一般是 MATLAB 安装目录的子目录 work 中），这样就不用设置 m 文件的路径了，否则，应当用路径操作指令 path 重新设置路径。另外，m 文件名不应该与 MATLAB 的内置函数名及工具箱中的函数重名，以免发生执行错误命令的现象。

MATLAB 对命令文件的执行等价于从命令窗口中顺序执行文件中的所有指令。命令文件可以访问 MATLAB 工作空间里的任何变量及数据。命令文件运行过程中产生的所有变量都等价于从 MATLAB 工作空间中创建这些变量。因此，任何其他命令文件和函数都可以自由地访问这些变量，这些变量一旦产生就一直保存在内存中，只有对其重新赋值，原有值才会变化。关机后，变量也就全部消失了。另外，在命令窗口中运行 clear 命令，也可以把这些变量从工作空间中删去。当然，在 MATLAB 的工作空间窗口中也可以用鼠标选择想要删除的变量，从而将这些变量从工作空间中删除。

M 文件编辑器一般不会随着 MATLAB 的启动而启动，只是用户在通过命令将其打开时，该编辑器才启动。需要指出的是，M 文件编辑器不仅可以用来编辑 M 文件，还可以对 M 文件进行交互式调试。而且，M 文件编辑器还可以用来阅读和编辑其他的 ASCLL 码

文件。通常情况下，可以使用下面几种方法打开 M 文件编辑器。

- 单击常用工具栏上的“新建”图标。
- 单击【File】/【New】/【M-File】菜单命令新建空白 M 文件。
- 可以在“命令”窗口中直接输入 edit 命令，或使用 edit mfiles 命令编辑某个已经存在的 M 文件，其中，mfiles 为用户需要编辑的文件名(可以不带扩展名)，如果是第一次创建 M 文件，系统会将名字设置为“Untitled”(未命名)的。如图 5-1 所示。

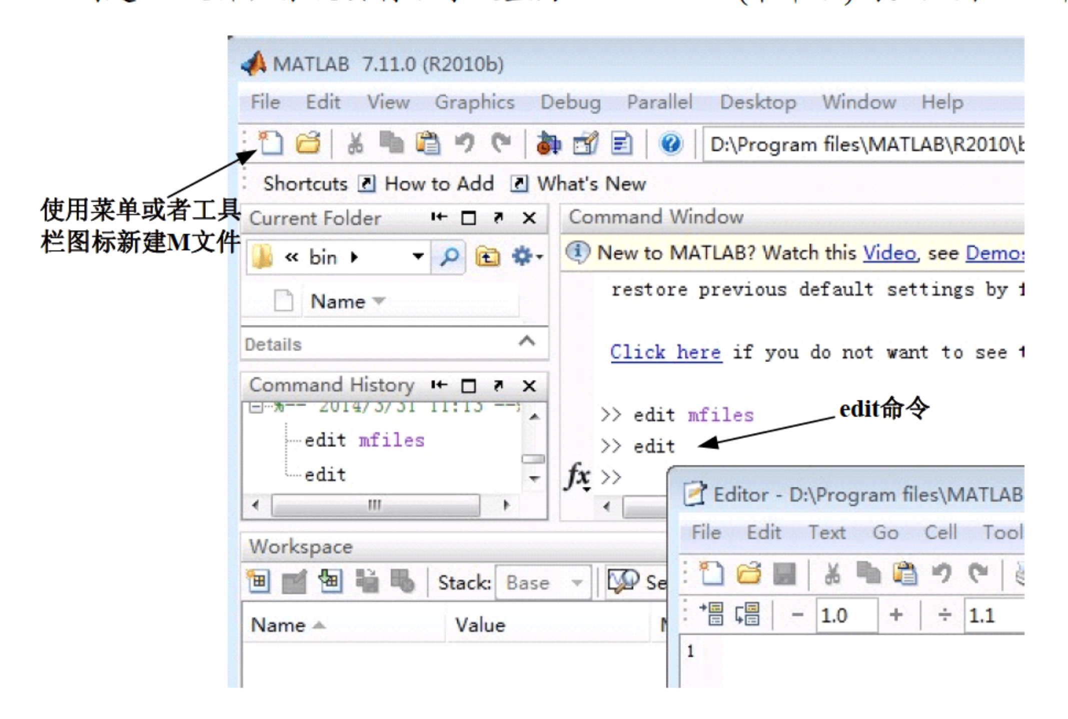


图 5-1　打开 M 文件编辑器

通过以上方法就可以打开 M 文件，如图 5-2 所示。

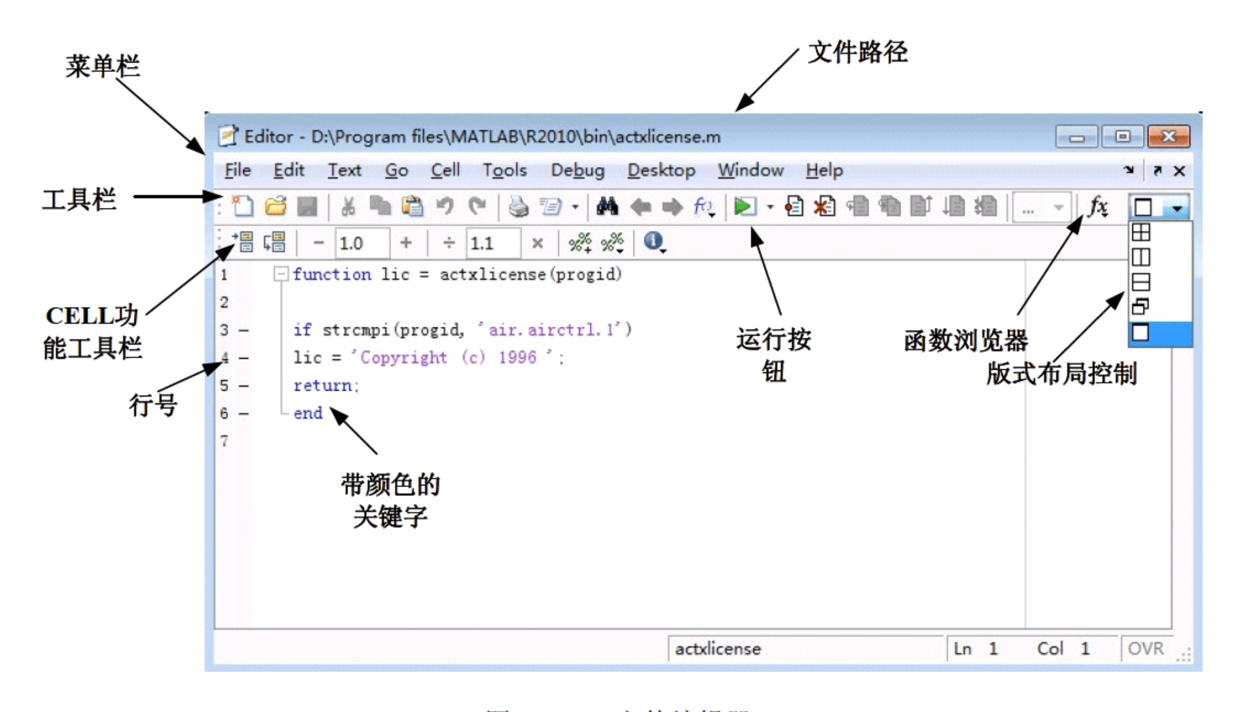


图 5-2　M 文件编辑器

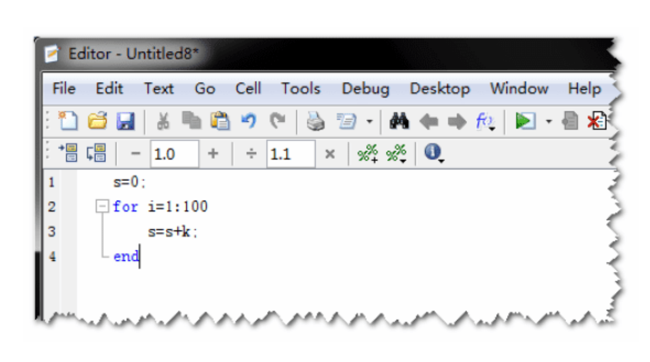
图 5-2 中对 M 文件编辑器的主要内容进行了标注。可以看出 M 文件编辑器的功能是非常多的。需要指出的是，有很多功能是最新版本的 MATLAB 才有的，这也是建议读者尤其是新手使用新版的 MATLAB 的原因之一。

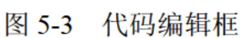
下面编写一个 M 文件，以 test.m 为例，用来计算矩阵 1 到 100 的和，并把它放到变量 s 中。

（1）创建新的 M 文件。单击常用工具栏上的“新建”图标。

（2）编写代码。在接下来出现的编辑框中输入相应的代码，如图 5-3 所示。

（3）保存。利用编辑框中的菜单【File】/【Save】命令，或者直接单击其上的图标，弹出一个保存文件的对话框（最好保存在自己熟悉的地方，以方便查找），文件名为 test，如图 5-4 所示。




图 5-3　代码编辑框

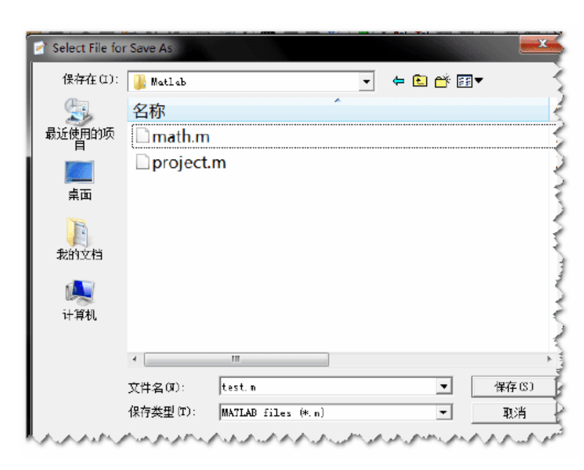


图 5-4　保存文件

（4）M 文件的使用。回到 MATLAB 的主界面，在命令窗口输入如下两条命令：

```
>>test
>>s
```

观察结果，如图 5-5 所示。

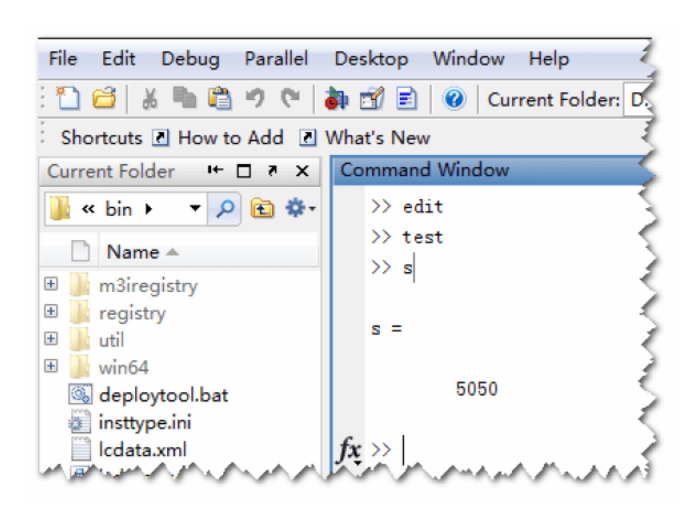


图 5-5　显示结果

5.1.2 M 文件的打开

上面已经创建了一个 M 文件，名为 test.m。打开 M 文件的常见方法如下。

- ❑ 通过【File】/【Open】命令找到对应的 M 文件即可。
- ❑ 通过 edit mfiles 命令编辑某个已经存在的 M 文件，其中 mfiles 为用户需要编辑的文件名(可以不带扩展名)，如 edit test.m。
- ❑ 单击工具栏上的按钮也可以打开对应的 M 文件。

5.1.3 M 文件的基本内容

【例 5-1】 简单函数 M 文件示例

本例以一个求 n 的阶乘的函数 M 文件为例，简单介绍 M 文件的基本单元，代码如下。

```
fact.m
function f = fact(n)                      %函数定义行，脚本式 M 文件无此行
% Compute a factiorrial value.            %H1 行
% FACT(N) returns the factional of N,     %Help 文本
% usually denoted by N!
% Put simple,Fact(N) is PROD(1:N).        %注释
f=prod(1:n);                              %函数体或脚本主体
```

在 fact.m 文件中，包含了一个 M 文件所包含的基本内容。M 文件的基本内容如表 5-1 所示。

表 5-1 M 文件的基本内容

M 文件内容	说　明
函数定义行（只存在于函数文件）	定义函数名称，定义输入输出变量的数量，顺序
H1 行	对程序进行的一行总结说明
Help 文本	对程序的详细说明，在调用 help 命令查询此 M 文件时和 H1 行一起显示在命令窗口
M 文件内容	说明
注释	具体语句的功能注释、说明
函数体	进行实际计算的代码

1. 函数定义行

函数定义行被用来定义函数名称，定义输入输出变量的数量、顺序。注意脚本式 M 文件没有此行。完整的函数定义语句为

```
function [outl,out2,out3...]=funName(in1, in2, in3...)
```

其中，输入变量用圆括号，变量间用英文逗号“,”分隔。输出变量用方括号，无输出可用空括号[]，或无括号和等号。无输出的函数定义行可以为

```
function funName(inl,in2,in3...)
```

在函数定义行中，函数的名字所能够允许的最大长度为 63 字符，个别操作系统有所

不同，用户可自行使用 namelengthmax 函数查询系统允许的最长文件名。另外，函数文件保存时，MATLAB 会默认以函数的名字来保存，请不要更改此名称，否则调用所定义的函数时会发生错误，不过脚本文件并不受此约束。funName 的命名规则与变量命名规则相同，不能是 MATLAB 系统自带的关键词，不能使用数字开头，也不能包含非法字符。

2. H1 行

H1 行紧跟着函数定义行，因为它是 help 文本的第一行，所以叫它 H1 行。用百分号(%)开始。MATLAB 可以通过命令把 M 文件上的帮助信息显示在命令窗口。因此，建议写 M 文件时建立帮助文本，把函数的功能、调用函数的参数等描述出来，以供自己和别人查看，方便函数的使用。

H1 行是函数功能的概括性描述，在命令窗口提示输入命令可以显示 H1 行文本：

```
help filename 或 lookfor filenam e
```

3. help 文本

这是为帮助建立的文本，可以是连续多行的注释文本。只能在命令窗口观看，不可以在 MATLAB Help 浏览器中显示。帮助文本遇到之后的第 1 个非注释行结束，函数中的其他注释行不被显示。

例如【5-1】中的 function f = fact(n)函数，可以将其保存在当前目录下，并且文件名为 fact.m，在命令行中调用 help 函数就可以看到相应的帮助文本。

【例 5-2】 help 文本查看示例。

在本例中，将演示通过 help 命令查看 M 文件中的帮助文本的过程。

```
>>help fact
   Compute a factiorrial value.              %H1 行
FACT(N) returns the factional of N,          %Help 文本
usually denoted by N!
```

以上命令的结果显示了 fact 函数文件的注释行，直到第 1 个非注释行——空行。

键入 lookfor 命令可见如下内容。

```
>>lookfor fact
fact                         - Compute a factiorrial value.
factor                       - Prime factors.
factorial                    - Factorial function.
chol                         - Cholesky factorization.
cholupdate                   - Rank 1 update to Cholesky factorization.
…
```

lookfor 命令是在所有命令中搜索包含 fact 字符串的函数，将这些函数列出来，并且将它们的 H1 行显示出来。从 lookfor 命令的结果中，可以看到 4 个其他的包含 fact 字符串的函数以及要找的 fact 函数，并且分别显示了它们的 Hl 行。

4. 注释

以%开始的注释行可以出现在函数的任何地方，当然也可以出现在一行语句的右边。

若注释行很多，可以使用注释块操作符“%{”和“%}”，下面给出一个简单的实例演

示注释块操作符。

【例 5-3】 注释块操作符示例。将【例 5-1】的 fact 函数中的多行注释改写为注释块。

```
function f = fact(n)                    %函数定义行，脚本式 M 文件无此行
%{
Compute a factiorrial value.            %H1 行
FACT(N) returns the factional of N,     %Help 文本
usually denoted by N!
%}
Put simple,Fact(N) is PROD(1:N).        %注释
f=prod(1:n);                            %函数体或脚本主体
```

将多行注释改为注释块并不影响运行结果。注释行和注释块的作用就是对程序进行注释，方便以后进行阅读和维护，程序运行时是不会运行注释的。

5. 函数体

函数体是函数和脚本中计算和处理数据的主体，可以包含进行计算和赋值的语句、函数调用、循环和流控制语句，以及注释语句、空行等。

5.1.4 M 文件的分类

M 文件有两大类：M 脚本文件（M-file Scripts）和 M 函数文件（M-file Functions）。

M 文件命名时注意以下要点。

- M 文件名的命名要符合“变量名命名规则”。MATLAB 的 isvarname 指令可检查用户所起文件名是否符合此规则。
- 除非特殊需要，用户应保证自己所创建的 M 文件名称具有唯一性。要避免与 MATLAB 所提供的函数同名。MATLAB 的 which 指令能帮助用户检查 M 文件名的唯一性。比如，用户想采用 filter 作为自己的文件名，那么可在 MATLAB 指令窗中运行以下指令，若在 MAYLAB 搜索路径上已存在 filter 命名的 M 文件，那么用户不应再采取此名。

```
which -all test
D:\ProgramFiles\MATLAB\R2010b\toolbox\stats\stats\@classregtree\test.m %
classregtree method
```

1. M 脚本文件

1）一般性说明

当指令窗中运行指令越来越多，控制流复杂度增加，或需要重复运行相关指令时，再从指令窗直接输入指令进行计算就显得烦琐，此时使用 M 脚本文件最适宜。

M 脚本文件的构成比较简单，其特点如下。

- 它是一串按用户意图排列而成的(包括控制流向指令在内的)MATLAB 指令集合。
- 脚本文件运行后，产生的所有变量都驻留在 MATLAB 基本工作空间(Base workspace)中。只要用户不使用 clear 指令加以清除，且 MATLAB 指令窗不关闭，这些变量将一直保存在基本工作空间中。基本工作空间随 MATLAB 的启动而产生；只有当

关闭 MATLAB 时，该基本工作空间才被删除。

2）M 脚本文件的基本结构

❑ 由%号起首的 H1 行（The first held text line），包括文件名和功能简述。
❑ 以%开头的在线帮助文本(help text)区：H1 行及其之后的所有连续注释行构成整个在线帮助文本，其涉及文件中关键变量的简短说明。
❑ 编写和修改记录：该区域文本内容也都以%开头；标志编写及修改该 M 文件的作者、日期和版本记录，可用于软件档案管理。
❑ 程序体(附带关键指令功能注解)。在 M 文件中，由%号引领的行或字符串都是“注解说明”，在 MATLAB 中不被执行。

2. M 函数文件

1）一般性说明

与脚本文件不同。函数文件（Function file）犹如一个“黑箱”。从外界只能看到传给它的输入量和送出来的计算结果，而内部运作可以藏而不见，其特点如下。

❑ 从形式上看，与脚本文件不同，函数文件的第一行总是以“function”引导的“函数声明行”(Function declaration line)，该行还罗列出函数与外界交换数据的全部“标称”输入输出量。输入输出量的“数目”并没有限制，既可以完全没有输入输出量，也可以有任意数目的输入输出量。
❑ MATLAB 允许使用比“标称数目”少的输入输出量，实现对函数的调用。
❑ 从运行上看，与脚本文件运行不同，每当函数文件运行，MATLAB 就会专门为其开辟一个临时工作空间，该空间称之为函数工作空间(Function workspace)。所有中间变量都存放在函数工作空间中。当执行完文件最后一条指令后，或遇到 return，就结束该函数文件的运行，同时该临时函数空间及其所有的中间变量立即被清除。
❑ 函数空间随具体 M 函数文件被调用而产生，随调用结束而删除，函数空间是相对的。
❑ 基本空间是独立的、临时的。在 MATLAB 整个运行期间，可以产生任意多个临时函数空间。
❑ 假如在函数文件中，发生对某脚本文件的调用，那么，该脚本文件运行产生的所有变量都存放于那函数空间之中，而不是存放在基本空间。

2）M 函数文件的基本结构

❑ **函数声明行(Function Declaration Line)**

它位于函数文件的首行，以 MATLAB 关键字 function 开头，函数名及函数的输入/输出量名都在这一行被定义。

❑ **H1 行**

紧随函数声明行之后以%开头的第一注释行。按 MATLAB 自身文件的规则，H1 行包含函数文件名，运用关键词简要描述该函数功能。

Hl 行提供 lookfor 关键词查询和 help 在线使用帮助。顺便指出，MATLAB 自带的函数文件在此行中都把函数文件名用“大写英文字母”表达。但实际上，此文件的“文件保存名”及运行时的“文件调用名”都必须是“相应的小写英文字母”。

❑ **在线帮助文本(Help Text)区**

H1 行及其之后连续的以%开头的所有注释行构成整个在线帮助文本。它通常包括函数输入输出量的含义及调用格式说明。

H1 行尽量使用英文表达，以便借助 lookfor 进行“关键词”搜索。但从 MATLAB 7.x 版起，lookfor 已经支持中文搜索，所以，H1 行现也可采用中文描述。

❑ **编写和修改记录**

其几何位置与在线帮助文本相隔一个“空行(不用%符开头)”。

该区域文本内容也都以%开头；标志编写及修改该 M 文件的作者和日期；版本记录。它用做软件档案管理。

❑ **函数体(Function Body)**

为清晰起见，与前面的注释可以用“空”行相隔，这部分内容由实现该 M 函数文件功能的 MATLAB 指令组成。它接受输入量，进行程序流控制，创建输出量。其中为阅读、理解方便，也配置适当的空行和注释。

若仅从运算角度看，唯“函数声明行”和“函数体”两部分是构成 M 函数文件所必不可少的。

【例 5-4】 分别编写出求取平均值与标准差的脚本文件 stat1.m 和函数文件 stat 2.m。

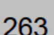
（1）单击 MATLAB 命令窗口工具栏中的按钮，打开 M 文件编辑器，在 M 文件编辑器中输入以下程序代码。

```
stat1.m  %脚本文件
%求阵列 x 的平均值和标准差
[m,n]=size(x);
if m==1
    m=n;
end
s1=sum(x);
s2=sum(x.^2);
mean1=s1/m;
stdev=sqrt(s2/m-mean1^2);

stat2.m   %函数文件
function [mean1,stdev]=stat2(x)
%STAT2 函数文件
%求阵列 x 的平均值和标准差
[m,n]=size(x);
if m==1
    m=n;
end
s1=sum(x);
s2=sum(x.^2);
mean1=s1/m;
stdev=sqrt(s2/m-mean1^2);
```

（2）在命令窗口依次输入如下命令。

```
>>clear
>>x=rand(4,4)+2;
```

```
    >>stat1         %执行 stat1.m 后，观察基本空间中的变量情况
    whos            %可见：脚本文件所产生的所有变量都返回了工作空间
    Name        Size          Bytes Class
      m          1x1           8     double array
      mean1      1x4           32    double array
      n          1x1           8     double array
      s1         1x4           32    double array
      s2         1x4           32    double array
      stdev      1x4           32    double array
      x          4x4           128   double array
>>disp([mean1;stdev])      %观察计算结果
2.7891    2.3084    2.2860    2.3083
0.2192    0.3196    0.1852    0.2699
```

（3）在命令窗口依次输入如下命令。

```
>>clear m n s1 s2 mean1 stdev
>>[m1,std1]=stat2(x);  %执行 stat2.m 后，观察其基本空间中的变量情况
>>whos                 %只增加了由函数返回的结果
  Name       Size          Bytes Class
  m1         1x4           32 double array
  std1       1x4           32 double array
  x          4x4           128 double array
>>disp([m1;std1])      %观察计算结果，和 stat1.m 一致
2.7891    2.3084    2.2860    2.3083
0.2192    0.3196    0.1852    0.2699
```



- 运行脚本文件，产生的所有变量都驻留在 MATLAB 基本工作空间(Base workspace)，只要不使用 clear 且不关闭指令窗口，这些变量将一直保存。(基本工作空间随 MATLAB 的启动而产生，只有关闭 MATLAB 时，该基本空间才被删除。)
- 运行函数文件，MATLAB 就会专门开辟一个临时工作空间，称为函数工作空间(Function workspace)，所有中间变量都存放在函数工作空间中，当执行完最后一条指令或遇到 return 时，就结束该函数文件的运行，同时该临时函数工作空间及其所有中间变量立即被清除。(函数工作空间随具体 M 函数文件的被调用而产生，随调用结束而删除。函数工作空间是相对基本工作空间对立的、临时的。在 MATLAB 这个运行期间，可以产生任意多个临时函数工作空间，而非基本工作空间。)
- 如果在函数文件中，调用了某脚本文件，那么，该脚本文件运行所产生的所有变量都放在该函数工作空间中，而不是放在基本工作空间中。

5.2 与外部数据的交换

MATLAB 应用中常需要实现 MATLAB 与其他应用程序的数据共享，即需将数据文件读入 MATLAB 进行有效的数据处理，然后将 MATLAB 处理好的数据保存为数据文件，以便其他应用程序所使用。MATLAB 支持多种文件格式的输入和输出，如.dat、.txt、.mat、.bmp 等。

5.2.1　数据的基本操作

1．数据文件保存

MATLAB 支持工作区的保存。用户可以将工作区或工作区中的变量以文件的形式保存，以备在需要时再次导入。保存工作区可以通过菜单进行，也可以通过命令窗口进行。

1）保存整个工作区

选择 File 菜单中的 Save Workspace As…命令，或者单击工作区浏览器工具栏中的 Save，可以将工作区中的变量保存为 MAT 文件。

2）保存工作区中的变量

在工作区浏览器中，右击需要保存的变量名，选择【Save As…】命令，将该变量保存为 MAT 文件。

3）利用 save 命令保存

该命令可以保存工作区，或工作区中任何指定文件，该命令的调用格式如下。

- **save('filename')**　将工作区中的所有变量保存为文件，文件名由 filename 指定。如果 filename 中包含路径，则将文件保存在相应目录下，否则，默认路径为当前路径。
- **save('filename', 'var1', 'var2', ...)**　保存指定的变量在 filename 指定的文件中。
- **save('filename', '-struct', 's')**　保存结构体 s 中全部域作为单独的变量。
- **save('filename', '-struct', 's', 'f1', 'f2', ...)**　保存结构体 s 中的指定变量。
- **save('-regexp', expr1, expr2, ...)**　通过正则表达式指定待保存的变量需满足的条件。
- **save('..., 'format')**　指定保存文件的格式，格式可以为 MAT 文件、ASCII 文件等。

2．数据文件的导入

MATLAB 中导入数据通常由函数 load 实现，该函数的用法如下。

- **load**　如果 matlab.mat 文件存在，导入 matlab.mat 中的所有变量，如果不存在，则返回 error。
- **load filename**　将 filename 中的全部变量导入到工作区中。
- **load filename X Y Z ...**　将 filename 中的变量 X、Y、Z 等导入到工作区中，如果是 MAT 文件，在指定变量时可以使用通配符“*”。
- **load filename -regexp expr1 expr2 ...**　通过正则表达式指定需要导入的变量。
- **load -ascii filename**　无论输入文件名是否包含有扩展名，将其以 ASCII 格式导入；如果指定的文件不是数字文本，则返回 error。
- **load -mat filename**　无论输入文件名是否包含有扩展名，将其以 mat 格式导入；如果指定的文件不是 MAT 文件，则返回 error。

【例 5-5】　将文件 matlab.mat 中的变量导入到工作区中。

（1）新建一个 matlab.mat 文件。打开 matlab，单击左上角文件（File），然后单击新建（new），选择变量（Variable），即可打开一个编辑器，输入数据即可。程序如下。

```
%生成基础测量数据
x=-3*pi:3*pi;
y=x;
```

```
[X,Y]=meshgrid(x,y);
R=sqrt(X.^2+Y.^2)+eps;
Z=sin(R)./R;
[dzdx,dzdy]=gradient(Z);
dzdr=sqrt(dzdx.^2+dzdy.^2);
%绘制基础数据图形
surf(X,Y,Z,abs(dzdr))
colormap(spring)
alphamap('rampup')
colorbar
%进行二维插值运算
xi=linspace(-3*pi,3*pi,100);
yi=linspace(-3*pi,3*pi,100);
[XI,YI]=meshgrid(xi,yi);
ZI=interp2(X,Y,Z,XI,YI,'cubic');
%绘制插值后的数据图形
figure
surf(XI,YI,ZI)
colormap(spring)
alphamap('rampup')
colorbar
```

（2）首先应用命令 whos －file 查看该文件中的内容。

```
>>  whos -file matlab.mat
  Name            Size            Bytes     Class    Attributes
  R               19x19            2888     double
  X               19x19            2888     double
  XI              100x100         80000     double
  Y               19x19            2888     double
  YI              100x100         80000     double
  Z               19x19            2888     double
  ZI              100x100         80000     double
  dzdr            19x19            2888     double
  dzdx            19x19            2888     double
  dzdy            19x19            2888     double
  s               -                 112     sym
  t               -                 112     sym
  tao             -                 112     sym
  unnamed         1x1                 8     double
  x               1x19              152     double
  xi              1x100             800     double
  y               1x19              152     double
  yi              1x100             800     double
```

（3）将该文件中的变量导入到工作区中。

```
>> load matlab.mat
```

（4）该命令执行后，可以在工作区浏览器中看见这些变量，如图 5-6 所示。

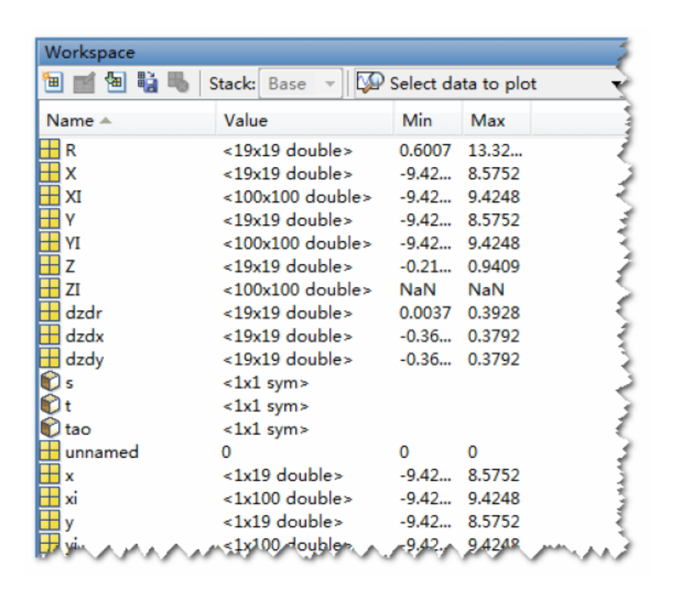


图 5-6 导入变量后的工作区视图

（5）用户可以访问这些变量

```
>> dzdy
dzdy =
     Columns 1 through 10
-0.0459 -0.0568     -0.0418 -0.0065 0.0353  0.0706  0.0922  0.1008
 0.1014  0.1001
-0.0483 -0.0463     -0.0195 0.0206  0.0580  0.0808  0.0863  0.0795
 0.0689      0.0620
-0.0432 -0.0180     0.0238  0.0617  0.0785  0.0688  0.0399  0.0053
 -0.0228     -0.0375
-0.0232 0.0153      0.0543  0.0701  0.0517  0.0062  -0.0477  -0.0921
 -0.1190     -0.1300
…
```

MATLAB 中，另一个导入数据的常用函数为 importdata，该函数的用法如下。

- **importdata('filename')** 将 filename 中的数据导入到工作区中。
- **A = importdata('filename')** 将 filename 中的数据导入到工作区中，并保存为变量 A。
- **importdata('filename','delimiter')** 将 filename 中的数据导入到工作区中，以 delimiter 指定的符号作为分隔符。

【例 5-6】从文件中导入数据。

```
 >>  imported_data = importdata('matlab.mat')
imported_data =
    unnamed: 0
          t: [1x1 sym]
        tao: [1x1 sym]
         …
         YI: [100x100 double]
         ZI: [100x100 double]
```



与 load 函数不同，importdata 将文件中的数据以结构体的方式导入到工作区中。

3．数据文件的打开

MATLAB 中可以使用 fopen 命令打开磁盘中各种格式的文件，MATLAB 根据文件的扩展名自动选择相应的编辑器。

【例 5-7】 在 MATLAB 中使用 fopen 命令打开磁盘文件。

（1）以读写的方式打开磁盘文件 fgetl.m。在 MATLAB 的命令窗口中输入以下代码。

```
>> [fid,message] = fopen('fgetl.m','r+')
```

（2）查看程序结果。在输入以上代码后，得到的结果如下。

```
fid =
    -1
message =
        No such file or directory
```



要注意当前 M 文件保存的位置，否则，会出现错误信息！

（3）查看 M 文件的程序代码。为了和后面步骤中打开的文件内容相比较，下面列出该文件的代码。

```
function tline = fgetl(fid)
try
[tline,lt] = fgets(fid);
tline = tline(1:end-length(lt));
if isempty(tline)
    tline = '';
end
catch
if nargin ==1
    error(nargchk(1,1,nargin,'struct'))
end
if isempty(fopen(fid))
    error('MATLAB:fgetl:invalidFID','Invalid file identifier.')
end
rethrow(lasterror)
end
```

尽管在系统中存在该函数文件，但是当该文件不在搜索路径上，fopen 以读写方法打开时，将会返回错误信息。

（4）以只写的方式打开磁盘文件 fgetl.m。在 MATLAB 的命令窗口中输入以下代码。

```
>> [fid,message] = fopen('fget1.m','w')
```

（5）查看程序结果。以上程序代码得到的结果如下。

```
fid =
 4
message =
    ''
```

前面步骤已经提到，fgetl. m 并不在命令搜索路径上，但是该命令并没有返回错误信息，

而是返回了正整数的信息，表示已经打开该文件，这是因为当以只写方式打开文件时，如果命令没有搜索到对应的文件，则会自动创建该文件。因此，当用户使用该命令后，系统会在搜索路径\MATLAB\work 上创建一个空白的 M 文件，该文件名称为 fgetl。

这些例子基本演示了 MATLAB 中 fopen 命令的使用方法，其对应的完整调用格式如下。

- [fid,message]=fopen(filename, mode)
- [fid,message]=fopen(filename, mode, machineformat)

在以上命令中，filename 表示的是打开文件的名称，mode 表示打开文件的方式，其具体包括以下类型。

- “r” 以只读方式打开文件；
- “w” 以只写方式打开文件，并覆盖原来的内容；
- “a” 增补文件，在文件尾部增加数据；
- “r+” 读写文件；
- “w+” 创建个新文件或者删除已有的文件内容，并进行读写操作；
- “a+” 读取和增补文件。

在默认情况下，MATLAB 会选择使用二进制的方式打开文件，而在该方式下，字符串不会被特殊处理。如果需要用文本形式打开文件，则应在以上 mode 字符串后面添加“t”，如“rt”、“rt+”等。

在两种 fopen 命令格式中，fid 是一个非负整数，一般被称为文件标识，在 MATLAB 中，用户对文件的任何操作，都需要通过 fid 参数来传递，MATLAB 会根据 fid 的数值来标识所有已经打开的文件，然后实现对文件的读、写和关闭等各种操作。如果程序代码得到 fid 的数值是–1，则表示 fopen 不能打开对应的文件，可能是因为该文件本身不存在，用户却以读写的方式打开，或者文件存在但是不在搜索路径上。

注意 open('filename.mat')和 load('filename.mat')的不同，前者将 filename.mat 以结构体的方式打开在工作区中，后者将文件中的变量导入到工作区中，如果需要访问其中的内容，需要以不同的格式进行。

【例 5-8】 open 与 load 的比较。

```
>> clear
>> A = magic(3);
>> B = rand(3);
>> save
Saving to: C:\Users\lyz\Desktop\matlab.mat
>> clear
>> load('matlab.mat')
>> A
A =
     8     1     6
     3     5     7
     4     9     2
>> B
B =
    0.9649    0.9572    0.1419
    0.1576    0.4854    0.4218
```

```
    0.9706    0.8003    0.9157
>> clear
>> open('matlab.mat')
ans =
    A: [3x3 double]
    B: [3x3 double]
>> struc1=ans;
>> struc1.A
ans =
     8     1     6
     3     5     7
     4     9     2
>> struc1.B
ans =
    0.9649    0.9572    0.1419
    0.1576    0.4854    0.4218
    0.9706    0.8003    0.9157
```

4. 数据文件的关闭

在打开文件后，如果完成了对应的读写工作，应该关闭文件，否则，打开的文件过多，造成系统资源的浪费。在本小节中将以一个简单的实例说明如何在 MATLAB 中关闭对应的文件。

【例 5-9】 在 MATLAB 中关闭对应的磁盘文件。

（1）创建文件 fgetl.m，然后删除该文件。在 MATLAB 的命令窗口中输入以下代码。

```
>> [fid,message] = fopen('fget1.m','w')
>> delete fgetl.m
```

（2）查看程序代码的结果。以上程序代码可以得到如下结果。

```
Warning: File not found or permission denied
```

以上结果表明，当用户使用 fopen 命令创建了对应的空白文件 fgetl.m，并打开对应的文件后，如果在关闭该文件前，直接使用删除文件，系统会提示用户删除命令被拒绝。

（3）首先关闭文件，然后删除该文件。在 MATLAB 的命令窗口中输入以下代码。

```
>> status = fclose(fid);
>> delete fgetl.m;
>> [fid,message] = fopen('fget1.m','r+');
```

（4）查看程序代码的结果。当用户输入以上程序代码后，得到的结果如下。

```
fid =
    -1
message =
        No such file or directory
```

从以上结果可以看出，当用户首先关闭对应的文件后，删除创建的文件，然后再次打开对应的文件，返回的信息是无法找到文件，表明已经删除了该文件。

以上例子已经演示了如何在 MATLAB 中关闭文件，在 MATLAB 中，可以使用 fclose

命令关闭已经打开的文件，其具体的调用命令如下。

```
status = fclose(fid)
status = fclose('all')
```

在以上命令中，fid 表示使用 fopen 命令得到的文件标识参数，第二个命令表示使用命令删除所有已经打开的文件。如果使用该命令得到的结果 status=0，则表示关闭文件的操作成功，否则，得到的结果 status=-1。

5.2.2 数据文件调用

在上一节中介绍的函数和命令主要用于读写 mat 文件，而在应用中，需要读写更多格式的文件，如文本文件、word 文件、xml 文件、xls 文件、图像文件和音视频文件等。本节介绍文本文件（txt）的读写。其他文件的读写，用户可以参考 MATLAB 帮助文档。

MATLAB 中实现文本文件读写的函数如表 5-2 所示。

表 5-2 现文本文件读写的函数

函　数	功　能
csvread	读入以逗号分隔的数据
csvwrite	将数据写入文件，数据间以逗号分隔
dlmread	将以 ASCII 码分隔的数值数据读入到矩阵中
dlmwrite	将矩阵数据写入到文件中，以 ASCII 分隔
textread	从文本文件中读入数据，将结果分别保存
textscan	从文本文件中读入数据，将结果保存为单元数组

下面详细介绍这些函数。

1．csvread、csvwrite

csvread 函数的调用格式如下。

- **M = csvread('filename')** 将文件 filename 中的数据读入，并且保存为 M，filename 中只能包含数字，且数字之间以逗号分隔。M 是一个数组，行数与 filename 的行数相同，列数为 filename 列的最大值，对于元素不足的行，以 0 补充。
- **M = csvread('filename', row, col)** 读取文件 filename 中的数据，起始行为 row，起始列为 col，注意，此时的行列从 0 开始。
- **M = csvread('filename', row, col, range)** 读取文件 filename 中的数据，起始行为 row，起始列为 col，读取的数据由数组 range 指定，range 的格式为[R1 C1 R2 C2]，其中，R1、C1 为读取区域左上角的行和列，R2、C2 为读取区域右下角的行和列。

csvwrite 函数的调用格式如下。

- **csvwrite('filename',M)** 将数组 M 中的数据保存为文件 filename，数据间以逗号分隔。
- **csvwrite('filename',M,row,col)** 将数组 M 中的指定数据保存在文件中，数据由参数 row 和 col 指定，保存 row 和 col 右下角的数据。

- **csvwrite** 写入数据时每一行以换行符结束。另外，该函数不返回任何值。

2．dlmread、dlmwrite

dlmread 函数用于从文档中读入数据，其功能强于 csvread。dlmread 的调用格式如下。

- M = dlmread('filename')
- M = dlmread('filename', delimiter)
- M = dlmread('filename', delimiter, R, C)
- M = dlmread('filename', delimiter, range)

其中，参数 delimiter 用于指定文件中的分隔符，其他参数的意义与 csvread 函数中参数的意义相同，这里不再赘述。dlmread 函数与 csvread 函数的差别在于，dlmread 函数在读入数据时可以指定分隔符，不指定时默认分隔符为逗号。

dlmwrite 函数用于向文档中写入数据，其功能强于 csvwrite 函数，dlmwrite 函数的调用格式如下。

- **dlmwrite('filename', M)** 将矩阵 M 的数据写入文件 filename 中，以逗号分隔。
- **dlmwrite('filename', M, 'D')** 将矩阵 M 的数据写入文件 filename 中，采用指定的分隔符分隔数据，如果需要 tab 键，可以用“\t”指定。
- **dlmwrite('filename', M, 'D', R, C)** 指定写入数据的起始位置。
- **dlmwrite('filename', M, attribute1, value1, attribute2, value2, ...)** 指定任意数目的参数。
- **dlmwrite('filename', M, '-append')** 如果 filename 指定的文件存在，在文件后面写入数据，不指定时则覆盖原文件。
- **dlmwrite('filename', M, '-append', attribute-value list)** 叙写文件，并指定参数。

3．textread，textscan

当文件的格式已知时，可以利用 textread 函数和 textscan 函数读入。这里只介绍这两个函数应用的实例。

【例 5-10】 通过%读入文件，按照原有格式读取。

在“C:\Users\lyz\Desktop\Matlab\mat.txt”路径下保存文本文件，内容为 Sally Level1 12.34 45 Yes。

在命令窗口中输入：

```
>> [names, types, x, y, answer] = textread('C:\Users\lyz\Desktop\
Matlab\mat.txt', '%s %s %f %d %s',1)
names =
       'Sally'
    types =
         'Level1'
    x =
       12.3400
    y =
       45
    answer =
          'Yes'
```

【例 5-11】 在 MATLAB 中使用 csvread 和 dlmread 命令读取文本文件。

（1）查看原始的数据文件。在本实例中，需要读取的文件是 txtlist.dat，其文件中包含的主要数据如下。

```
02,04,06,08,10,12
03,06,09,12,15,18
05,10,15,20,25,30
07,14,21,28,35,42
11,22,33,44,55,66
```

（2）使用命令读取该数据文件。在 MATLAB 的命令窗口中输入以下程序代码。

```
>> m1 = csvread('txtlist.dat');
>> m2 = csvread('txtlist.dat',2,0);
>> m3 = csvread('txtlist.dat',2,0,[2,0,3,3]);
>> m4 = dlmread('txtlist.dat');
```

（3）查看程序代码的结果。在命令窗口中输入变量名称，得到的结果如下。

```
>> m1
m1 =
     2    4    6    8    10   12
     3    6    9    12   15   18
     5    10   15   20   25   30
     7    14   21   28   35   42
     11   22   33   44   55   66
>> m2
m2 =
     5    10   15   20   25   30
     7    14   21   28   35   42
     11   22   33   44   55   66
>> m3
m3 =
     5    10   15   20
     7    14   21   28
>> m4
m4 =
     2    4    6    8    10   12
     3    6    9    12   15   18
     5    10   15   20   25   30
     7    14   21   28   35   42
     11   22   33   44   55   66
```

从以上结果可以看出，使用 csvread 和 dlmread 命令，可以根据不同的条件来读取原始文本文件中的数据，其中在 dlmread 命令中，用户可以自行设置数据之间的分隔符。

【例 5-12】 在 MATLAB 中使用 textread 命令来读取文本文件。

（1）查看原始的数据文件。在本实例中，需要读取的文件是 txtlist2.txt，其文件中包含的第一行数据如下。

```
Sally Level1 12.34 45 Yes
```

（2）使用命令读取该数据文件。在 MATLAB 的命令窗口中输入以下程序代码。

```
>> [names,types,x,y,answer] = textread('txtlist2.txt','%s %s %f %d %s',1);
```

（3）查看程序代码的结果。在命令窗口中输入变量名称，得到的结果如下。

```
>> names
names =
      'Sally'
>> types
types =
      'Levell'
>> x
x =
   12.3400
>> y
y =
   45
>> answer
answer =
       'Yes'
```

从以上结果可以看出，在 textread 命令中，用户可以指定读取数据的格式，得到对应的输入结果。

【例 5-13】 在 MATLAB 中使用 textscan 命令读取文本文件。

（1）查看原始的数据文件。在本实例中，用户需要读取的文件是树 txtscan.dat，其文件中包含的数据如下。

```
Sally Level1 12.34 45 1.23e10 inf NaN  Yes
Joe   Level2 23.54 60 9e19 -inf 0.001 No
Bill  Level3 34.90 12 2e5 10 100 No
```

（2）使用命令读取该数据文件。在 MATLAB 的命令窗口中输入以下程序代码。

```
        >> fid = fopen('txtscan.dat');
        >> Cl = textscan(fid,'%s %s %f32 %d8 %u %f %f %s');
        >> fclose(fid);
```

（3）查看程序代码的结果。在命令窗口中输入变量名称，得到的结果如下。

```
>> whos Cl
  Name      Size            Bytes  Class    Attributes
  Cl        1x8             2053  cell
```

可以看出，使用 txtscan 命令得到的结果将会被存储在元胞数组中，该元胞数组包含的列数就是原始数据文件中使用分隔符隔开的数据列。

（1）查看 Cl 数组的结果。在命令窗口中输入以下程序代码。

```
        >> for i=1:8
        disp(Cl{i}');
        end
```

（2）查看程序代码的结果。输入代码后，按“Enter”键，得到的结果如下。

```
'Sally'     'Joe'       'Bill'
'Level1'    'Level2'    'Level3'
12.3400     23.5400     34.9000
45          60          12
4294967295  4294967295  200000
Inf         -Inf        10
NaN         0.0010      100.0000
'Yes'       'No'        'No'
```

（3）读取原始文件，并忽略第三列数据。在命令窗口中输入以下程序代码。

```
>> fid = fopen('txtscan.dat');
>> C2 = textscan(fid,'%7c %6s %*f %d8 %u %f %f %s');
>> fclose(fid);
```

（4）查看 C2 的属性。在命令窗口中输入以下程序代码。

```
>> whos C2
Name      Size            Bytes  Class    Attribute
C2        1x7              1605  cell
```

（5）查看 C2 数组的结果。在命令窗口中输入以下程序代码。

```
>> for i=1:7
disp(C2{i}');
end
```

（6）查看程序代码的结果。输入代码后.按“Enter”键，得到的结果如下。

```
'Sally'     'Joe'       'Bill'
'Level1'    'Level2'    'Level3'
45          60   12
4294967295  4294967295  200000
Inf         -Inf        10
NaN         0.0010      100.0000
'Yes'       'No'        'No'
```

从以上结果可以看出，当在 textscan 命令中使用 textscan (fid, '%7c %6s%*f %d8 %u %f %f %s')后，其中，%*f 所替代的对应数据列会被跳过，不被读入。

（1）仅仅读取原始文件的第一列数据。在命令窗口中输入以下程序代码。

```
>> fid = fopen('txtscan.dat');
>> names = textscan(fid,'%s%*[^\n]');
>> fclose(fid);
```

（2）查看 names 的属性。在命令窗口中输入以下程序代码。

```
>> whos names
  Name       Size            Bytes  Class    Attributes
  names      1x1               472  cell
```

（3）查看程序代码的结果。在命令窗口中输入以下程序代码。

```
>> B = names{1}'
B =
'Sally'    'Joe'    'Bill'
```

（4）消除原始第三列数据前面的标签。在命令窗口中输入以下程序代码。

```
>> fid = fopen('txtscan.dat');
>> C3 = textscan(fid,'%s Level%u8 %f32 %d8 %u %f %f %s');
>> fclose(fid);
```

（5）查看程序代码的结果。输入代码后，按 Enter 键，得到的结果如下。

```
>> whos C3
  Name      Size            Bytes  Class    Attributes
  C3        1x8              1684  cell
```

（6）查看 C3 的属性。在命令窗口中输入以下程序代码。

```
    'Sally'     'Joe'       'Bill'
    1           2           3
    12.3400     23.5400     34.9000
    45          60          12
    4294967295  4294967295  200000
    Inf         -Inf        10
    NaN         0.0010      100.0000
    'Yes'       'No'        'No'
```

从以上结果中可以看出，相对于原始的第二列数据，通过该命令得到的第二列数据清除了字符串“Level”，只留下了数值代码。

【例 5-14】 使用 csvwrite 命令向文本文件写入 MATLAB 的数据。

（1）将数据写入 csblist 文本文件中。在 MATLAB 的命令窗口中输入以下程序代码。

```
>> m=[3 6 9 12 15;5 10 15 20 25;7 14 21 28 35;11 22 33 44 55];
>> csvwrite('csvlist.dat',m)
>> type csvlist.dat
```

（2）查看程序代码的结果。输入代码后，按 Enter 键，得到的结果如下。

```
3,6,9,12,15
5,10,15,20,25
7,14,21,28,35
11,22,33,44,55
```

（3）将数据写入 csblist 文本文件中，并在数据列的前侧添加两个数据列。在命令窗口中输入以下代码。

```
>> csvwrite('csvlist.dat',m,0,2)
>> type csvlist.dat
```

（4）查看程序代码的结果。输入代码后，按 Enter 键，得到的结果如下。

```
,,3,6,9,12,15
,,5,10,15,20,25
,,7,14,21,28,35
,,11,22,33,44,55
```

（5）将数据写入 csblist 文本文件中，并在数据列的前侧添加 4 个数据列，同时在数据列上方添加 2 个数据行。在命令窗口输入以下代码。

```
>> csvwrite('csvlist.dat',m,2,4)
>> type csvlist.dat
```

（6）查看程序代码的结果。输入代码后，按 Enter 键，得到的结果如下。

```
,,,,,,,,
,,,,,,,,
,,,,3,6,9,12,15
,,,,5,10,15,20,25
,,,,7,14,21,28,35
,,,,11,22,33,44,55
```

在以上命令中，都是在没有对应数据文件之前使用写入命令，但是使用该命令会首先创建该文件，然后实现写入任务。

【例 5-15】 使用 dlmwrite 命令向文本文件写入 MATLAB 的数据。

（1）将数据写入 myfile 文本文件中。在 MATLAB 的命令窗口中输入以下程序代码。

```
>> m=rand(6);
>> dlmwrite('myfile.txt',m,'delimiter','\t','precision',5)
>> type myfile.txt
```

（2）查看程序代码的结果。输入代码后，按 Enter 键，得到的结果如下。

```
0.81472 0.2785  0.95717 0.79221     0.67874 0.70605
0.90579 0.54688 0.48538 0.95949     0.75774 0.031833
0.12699 0.95751 0.80028 0.65574     0.74313 0.27692
0.91338 0.96489 0.14189 0.035712    0.39223 0.046171
0.63236 0.15761 0.42176 0.84913     0.65548 0.097132
0.09754 0.97059 0.91574 0.93399     0.17119 0.82346
```

（3）修改数据精度，然后将数据写入 myfile 文本文件。输入以下程序代码。

```
>> m=rand(6);
>> dlmwrite('myfile.txt',m,'delimiter','\t','precision',3)
>> type myfile.txt
```

（4）查看程序代码的结果。输入代码后，按 Enter 键，得到的结果如下。

```
0.695   0.766   0.709   0.119   0.751   0.547
0.317   0.795   0.755   0.498   0.255   0.139
0.95    0.187   0.276   0.96    0.506   0.149
0.0344  0.49    0.68    0.34    0.699   0.258
0.439   0.446   0.655   0.585   0.891   0.841
0.382   0.646   0.163   0.224   0.959   0.254
```



（5）向 myfile 文本文件写入多行数据。输入以下程序代码。

```
>> M = ones(5);
>> dlmwrite('myfile.txt',[M*5 M/5],' ')
>> dlmwrite('myfile.txt',eye(4),'-append','roffset',1,'delimiter',' ')
>> type myfile.txt
```

（6）查看程序代码的结果。输入代码后，按“Enter”键，得到的结果如下。

```
5 5 5 5 5 0.2 0.2 0.2 0.2 0.2
5 5 5 5 5 0.2 0.2 0.2 0.2 0.2
5 5 5 5 5 0.2 0.2 0.2 0.2 0.2
5 5 5 5 5 0.2 0.2 0.2 0.2 0.2
5 5 5 5 5 0.2 0.2 0.2 0.2 0.2

1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

5.3 流程控制

MATLAB 的基本结构为顺序结构，即代码的执行顺序为从上到下。但是顺序结构远远不能满足程序设计的需要。为了编写更加实用、功能更加强大、代码更加精简的程序，需要使用流程控制语句。流程控制语句主要包括判断语句、循环语句、分支语句等。

5.3.1 顺序结构

顺序结构是最简单的程序结构，用户编写好程序之后，系统将按照程序的物理位置顺序执行，因此，这种程序比较容易编制。但由于它不包含其他的控制语句，程序结构比较单一，因此，实现的功能比较有限。尽管如此，对于比较简单的程序来说，使用顺序结构是能够很好地解决问题。

【例 5-16】 顺序结构示例。实现计算 a 与 b 的和与积，相乘后减去 c 的功能。编写 M 文件代码如下所示。

Ex_5_16.m

```
a=1;
b=2;
c=3;
ans1=a+b
ans2=a*b
ans3=ans2*ans1-c
```

单击 F5 快捷键或“运行”按钮，或者将其在当前目录下保存为 Ex_5_16.m，然后在命令窗口中键入 Ex_5_16.m 并运行，得到如下结果。

```
>> Ex 5 16
ans1 =
     3
ans2 =
     2
ans3 =
     3
```

【例 5-17】 顺序结构示例。在 MATLAB 中，使用顺序结构编写绘制函数的图形。

Ex_5_17.m

```
%定义符号变量 t 和 tao
syms t tao
%定义积分表达式
y=exp(-t/3)*cos(1/2*t);
%对表达式进行积分
s=subs(int(y,0,tao),tao,t);
%绘制积分图形
ezplot(s,[0,4*pi]);
grid
```

返回到 MATLAB 的命令窗口，输入 Ex_5_17，然后按 Enter 键，得到的结果如图 5-7 所示。

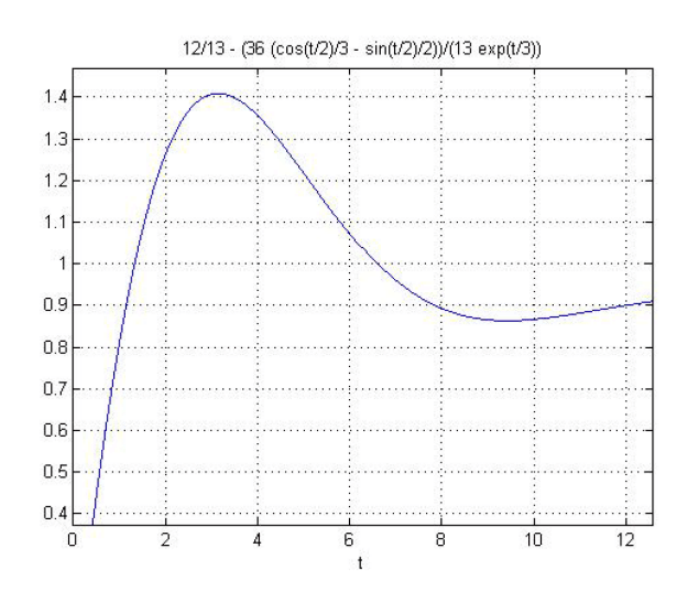
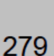


图 5-7　得到的程序结果

在上面的程序代码中，首先定义符号变量，然后定义积分表达式，进行积分运算，最后调用 ezplot 命令绘制积分函数的图形。这样的程序代码流程符合逻辑顺序，而且容易阅读，容易理解，这是顺序结构的重要优点。

5.3.2　选择结构

为了实现选择结构的程序，提供了 if 语句、switch 和 try-catch 语句。



1. if 语句

在编写程序时往往要根据一定的条件进行一定的判断，然后选择执行不同的语句，此时需要使用判断语句来进行流控制。

条件判断语句为 if…else…end，其使用形式有以下 3 种。

1）if…end

此时的程序结构如下。

```
if 表达式
    执行语句
end
```

这是最简单的判断语句，即当表达式为 true 时，则执行 if 与 end 之间的执行语句；当表达式为 false 时，则跳过执行语句，然后执行 end 后面的程序。

【例 5-18】 if…end 语句的实例。

EX_5_18.m

```
a=6;
if rem(a,2) == 0    %判断 a 是否是偶数
disp('a is even')
b=a/2
end
```

本例中的程序首先判断 a 是否是偶数，因为 a 的值为 6，所以命令 rem(a,2)==0 返回逻辑值 true。然后程序运行 if 语句之内的程序段，得出如下结果。

```
>> Ex 5 18
a is even
b =
    3
```

2）if…else…end

此时的程序结构如下。

```
if 表达式
执行语句 1
else
执行语句 2
end
```

如果表达式为 true，则执行 if 与 else 之间的执行语句 1，否则，执行 else 与 end 之间的执行语句 2。

【例 5-19 】 if…else…end 语句使用示例。

Ex_5_19.m

```
        if a >b
disp('a is bigger then b')          %a>b 则执行此句
y=a;
else
```

```
disp('a is not bigger then b')    %a<=b 则执行此句
y=b;
end
```

3）if…elseif…else…end

在有更多判断条件的情况下，可以使用 if…elseif…else…end 结构。

```
if 表达式 1
执行语句 1
elseif 表达式 2
执行语句 2
elseif 表达式 3
执行语句 3
elseif …
…
else
执行语句
end
```

在这种情况下。如果程序运行到的某一条表达式为 true，则执行相应的语句，此时系统不再对其他表达式进行判断，即系统将直接跳到 end，另外的 else 可有可无。

需要指出的是，如果 else 被空格或回车符分开，成为 else if，那么系统会认为这是一个嵌套的 if 语句，所以最后需要有多个 end 关键词相匹配，并不像 if…elseif…else…end 语句中那样只有一个 end 关键词。

【例 5-20】 if…elseif…else…end 语句使用示例。

```
if n < 0                         %如果 n 是负数，则显示错误信息
disp('Input must be positive');
elseif rem(n,2)==0               %如果 n 是偶数，则除以 2
A=n/2;
else
A=(n+1)/2;                       %如果 n 是奇数，则加 1，然后除以 2
End
```

在大多数情况下，条件表达式会由关系表达式或逻辑表达式组成，这些表达式返回的都是逻辑值 0 或 1，将作为条件判断的依据。为了提高程序代码执行的效率，MATLAB 会尽可能少地检测这些表达式的数值。

【例 5-21】 在 MATLAB 中，使用 if 分支结构编写求解一元二次方程 $ax^2+bx+c=0$ 的程序代码，并且运行检测该代码结果。

- ❑ 分析分支结构的判断条件。根据基础数学知识可知，一元二次方程 $ax^2+bx+c=0$ 的根的性质直接取决于判别式 $\Delta=b^2-4ac$ 的数值。当 $\Delta=0$ 时，该方程有两个相等的实根;当 $\Delta>0$ 时，该方程有两个互不相等的实根_当 $\Delta<0$ 时，该方程有两个虚根。
- ❑ 单击 MATLAB 命令窗口工具栏中的按钮。打开 M 文件编辑器，在 M 文件编辑器中输入以下程序代码。



calc_root.m

```
%script file cale root.m
%purpose:
%This program solves for the roots of a quadratic equation
%of the form a*x^2+b*x+c=0.It calcaulates the answers of
%roots the equation posseses.

%Define variables:
%a=    coefficient of x^2
%b     coefficient of x
%c     constant term
%x1    first root of the equation
%x2    second root of the equation

disp('This program solves for the roots of a quadratic equation');
disp('of the form a*x^2+b*x+c=0');
a=input('Enter the coefficient A:');
b=input('Enter the coefficient B:');
c=input('Enter the coefficient C:');
discriminant=b^2-4*a*c;
%如果判别式大于 0
%则根据二元方程的公式得出两个不同的实数解
if discriminant>0
   x1=(-b+sqrt(discriminant))/(2*a);
x2=(-b-sqrt(discriminant))/(2*a);
 %在命令窗口显示求解结果
   disp('This equation has two real roots');
   fprintf('x1=%f\n',x1);
fprintf('x2=%f\n',x2);
%当判别式等于 0，则返回两个相同的实数根
elseif discriminant==0
   x1=-b/(2*a);
   disp('This equation has two identical roots');
fprintf('x1=x2=%f\n',x1);
%当判别式小于 0，则返回两个须根
else
   real_part=-b/(2*a);
   image_part=sqrt(abs(discriminant))/(2*a);
   disp('This equation has two complex roots');
   fprintf('x1=%f+i%f\n',real part,image part);
   fprintf('x2=%f-i%f\n',real part,image part);
end
```

- ❑ 单击 M 文件编辑器中的“保存”按钮，将以上程序代码保存为“calc_root.m”。
- ❑ 返回到 MATLAB 的命令窗口，输入“calc_root”，然后按 Enter 键，根据程序代码的提示，依次输入方程的系数，得到的结果如下。

```
>> calc root
This program solves for the roots of a quadratic equation
of the form a*x^2+b*x+c=0
```

```
Enter the coefficient A:1
Enter the coefficient B:5
Enter the coefficient C:6
This equation has two real roots
x1=-2.000000
x2=-3.000000
>> calc root
This program solves for the roots of a quadratic equation
of the form a*x^2+b*x+c=0
Enter the coefficient A:2
Enter the coefficient B:3
Enter the coefficient C:4
This equation has two complex roots
x1=-0.750000+i1.198958
x2=-0.750000-i1.198958
>> calc root
This program solves for the roots of a quadratic equation
of the form a*x^2+b*x+c=0
Enter the coefficient A:1
Enter the coefficient B:4
Enter the coefficient C:4
This equation has two identical roots
x1=x2=-2.000000
```

最后，在使用 if 分支结构时，需要注意以下几个问题。

（1）if 分支结构是所有程序结构中最灵活的结构之一，可以使用任意多个 elseif 语句，但是只能有一个 if 语句和一个 end 语句。

（2）if 语句可以相互嵌套，可以根据实际需要将各个 if 语句进行嵌套。来解决比较复杂的实际问题。

2. switch 语句

在 MATLAB 语言中，除了上面介绍的 if…else…end 分支语句外，还提供另外一种分支语句形式，那就是 switch…case…end 分支语句。这可以熟悉 C 语言或其他高级语言的用户更方便地使用 MATLAB 的分支功能，其使用语句如下。

```
switch 开关语句
   case 条件语句 1
       执行语句
   case 条件语句 2
       执行语句 2
…
otherwise
       执行语句
end
```

在 switch 分支结构中，当某个条件语句的内容与开关语句的内容相匹配时，系统将执

行其后的语句；如果所有的条件语句与开关条件都不相符时，系统将执行 otherwise 后面的语句。和 C 语言不同的是，switch 语句中如果某一个 case 中的条件语句为 true，则其他的 case 将不会再继续执行，程序将直接跳至 switch 语句结尾。

【例 5-22】 switch…case…end 示例 1。

```
switch var
        case 1                                  %判断 var 是不是 1
            disp('1')
        case {2,3,4}                            %判断 var 是不是 2,3,4
            disp('2 or 3 or 4')
        case 5                                  %判断 var 是不是 5
            disp('5')
        otherwise                               %其他情况
            disp('something else')
end
```

【例 5-23】 switch…case…end 示例 2。

Ex_5_23.m

```
clear;
%划分区域：满分(100)，优秀(90-99)，良好(80-89)，及格(60-79)，不及格(<60)
for i=1:10;a{i}=89+i;b{i}=79+i;c{i}=69+i;d{i}=59+i;end;c=[d,c];
Name={' Jack','Marry','Peter',' Rose',' Tom'};
Mark={72,83,56,94,100};Rank=cell(1,5);    %3 个数组，且都是(1x5)维的
%创建一个含 5 个元素的构架数组 S，它有三个域
S=struct('Name',Name,'Marks',Mark,'Rank',Rank);
%根据学生的分数，求出相应的等级
for i=1:5
   switch S(i).Marks
   case 100
      S(i).Rank='满分';
   case a
      S(i).Rank='优秀';
   case b
      S(i).Rank='良好';
   case c
      S(i).Rank='及格';
   otherwise
      S(i).Rank='不及格';
   end
end
%将学生姓名，得分，等级等信息打印出来
disp(['学生姓名  ',' 得分  ','等级']);disp(' ')
for i=1:5;
 disp([S(i).Name,blanks(6),num2str(S(i).Marks),blanks(6),S(i).Rank]);
end;
```

运行结果为：

```
>> Ex_5_23
学生姓名      得分        等级
Jack          72          及格
Marry         83          良好
Peter         56          不及格
Rose          94          优秀
Tom          100          满分
```

3. try-catch

在 MATLAB 中，try-catch 结构主要用来对异常情况进行处理，其相应的语法结构如下。

```
try
statement                  %组命令 1 总被执行。若正确，则 跳出此结构
catch
statement                  %仅当组命令 1 出现错误，组命令 2 才被执行
end
```

在以上语法结构中，try 后面的命令语句会被执行，只有当这些语句执行过程中出现错误时，catch 控制语句就会捕获它，执行相应的语句。如果执行 catch 语句后的命令又出现错误，MATLAB 就会终止该程序结构。



- 可以用 lasterr 函数查询出错原因。如果函数 lasterr 的运行结果为一个空串，则表明组命令 1 被成功执行了。
- 当执行组命令 2 时又出错，MATLAB 将终止该结构。

【例 5-24】 try-catch 结构应用实例：对 (3×3)魔方阵的行进行援引，当“行下标”超出魔方阵的最大行数时，将改向对最后一行的援引，并显示“出错”警告。

（1）单击 MATLAB 命令窗口工具栏中的按钮，打开 M 文件编辑器，在 M 文件编辑器中输入以下程序代码。

Ex_5_24.m

```
clear
N=4;A=magic(3);                %设置 3 行 3 列矩阵 A
try
        A_N=A(N,:)             %取 A 的第 N 行元素
catch
        A_end=A(end,:)         %如果取 A(N,:)出错，则改取 A 的最后一行
end
lasterr                        %显示出错原因
```

（2）单击 M 文件编辑器中的“保存”按钮，将以上程序代码保存为“Ex_5_24.m”。

（3）返回到 MATLAB 的命令窗口，输入“Ex_5_24”，然后按 Enter 键，根据程序代码的提示，依次输入方程的系数，得到的结果如下。

>> Ex_5_24

```
A end =
     4     9     2
ans =
Attempted to access A(4,:); index out of bounds because size(A)=[3,3].
```

5.3.4 循环结构

MATLAB 控制程序流的关键词与其他编程语言十分相似。因此，本节对于各组关键词的用法描述比较简明，且大多通过算例进行。

1. for 循环和 while 循环控制

尽管，MATLAB 很适宜向量化编程，本书也一再强调采用向量化编程而尽量少用循环，但循环仍是数据流的基本控制手段，在许多应用场合仍不可完全避免。

循环结构的基本形式如下。

MATLAB 中的 for 循环和 while 循环的结构及其使用方式如表 5-3 所示。

【例 5-25】 for 循环使用示例 1。

EX_5_25.m

```
x=ones(1,6)
for n=2:6                %循环控制
    x(n) = 2*x(n-1)      %循环体
end
```

运行后可得到如下结果。

```
>>  Ex 5 25
x =
     1     1     1     1     1     1
x =
     1     2     1     1     1     1
x =
     1     2     4     1     1     1
x =
     1     2     4     8     1     1
x =
     1     2     4     8    16     1
x =
     1     2     4     8    16    32
```

表 5-3 循环结构的基本使用方式

for 循环	while 循环
for variable=initval:stepval:endval statement … statement end	while expression (commands) end

for 循环	while 循环
variable 表示变址。initval:stepval: endval 表示一个以 initval 开始，以 endval 结束，步长为 stepval 的向量。其中 initval-stepval 和 endval 可以是整数、小数或负数。但当 initval<endval 时，stepval 则必须为大于 0 的数；而当 initval>endval 时，stepval 则必须为小于 0 的数。表达式也可以为 initval:endval 的形式，此时，stepval 的默认值为 1，initval 必须小于 endval。另外，还可以直接将一个向量赋值给 variable，此时，程序进行多次循环直至穷尽该向量的每一个值。variable 还可以是字符串、字符串矩阵或由字符串组成的单元阵	执行每次循环时，只是“控制表达式(Controlling Expression)”expression 为真，即非 0，就是该循环体的 commends；反之，结束循环； while 循环的次数是不确定的

【例 5-26】 for 循环使用示例 2。

EX_5_26.m

```
for m=1:5
    for n=1:10
        A(m,n)=1/(m+n-1);       %使用循环体给变量 A 赋值
    end
end
```

运行后可得到如下结果。

```
>> Ex 5 26
>> A
A =
  Columns 1 through 8
    1.0000  0.5000  0.3333   0.2500   0.2000   0.1667   0.1429   0.1250
    0.5000  0.3333  0.2500   0.2000   0.1667   0.1429   0.1250   0.1111
    0.3333  0.2500  0.2000   0.1667   0.1429   0.1250   0.1111   0.1000
    0.2500  0.2000  0.1667   0.1429   0.1250   0.1111   0.1000   0.0909
    0.2000  0.1667  0.1429   0.1250   0.1111   0.1000   0.0909   0.0833

  Columns 9 through 10

    0.1111    0.1000
    0.1000    0.0909
    0.0909    0.0833
    0.0833    0.0769
    0.0769    0.0714
```

需要指出的是，MATLAB 由于是解释性语言，它对于 for 和 while 循环的执行效率并不高，所以用户应尽量使用 MATLAB 更为高效的向量化语言来代替循环。

【例 5-27】 while 循环使用示例。

Ex_5_27.m

```
i=1;
while i<10                  %i 小于 10 进行循环
```

```
    x(i)=i^3;                %循环体内的计算
    i=i+1;                   %表达式值的改变
end
```

运行 Ex_5_27.m 文件，可以得到如下结果。

Ex_5_27

```
>> x
x =
     1     8    27    64   125   216   343   512   729
>> i
i =

10
```

【例 5-28】 多种循环体的嵌套使用示例。

Ex_5_28.m

```
clear
clc
for i=1:1:6                            %行号循环，从 1 到 6
    j=6;
    while j>0                          %列号循环，从 6 到 1
        x(i,j)=i-j;                    %矩阵 x 的第 i 行第 j 列元素值为其行号的差
        if x(i,j)<0
            x(i,j)=-x(i,j);            %当 x(i,j)为负数时，取其相反数
        end
        j=j-1;
    end
end
```

运行 Ex_5_28.m 文件，可以得到如下结果。

```
>> x
x =

     0     1     2     3     4     5
     1     0     1     2     3     4
     2     1     0     1     2     3
     3     2     1     0     1     2
     4     3     2     1     0     1
     5     4     3     2     1     0
```

2. 辅助控制指令 continue、break 和 return 命令等

在使用 MATLAB 设计程序时，经常会遇到提前终止循环、跳出子程序、显示错误信息等情况，因此，还需要其他的控制语句来实现上面这些功能。在 MATLAB 中，对应的控制语句有 continue、break、return、echo 等，本节中将详细介绍这些控制语句。

continue 和 break 为用户编写循环控制提供了更大的自由度，它们的具体含义如下。

continue 在 for 或 while 循环中遇到该指令，执行下一次迭代，不管其后指令如何。

break 在 for 或 while 循环中遇到该指令，跳出该循环，不管其后指令如何。

使用 return 命令，能够使得当前正在调用的函数正常退出。首先对特定条件进行判断，然后根据需求，调用 return 语句终止当前运行的函数。

❑ 结束循环——continue 命令

在 MATLAB 中，该命令的功能是结束程序的循环语句，也就是跳过循环体中还没有执行的语句，其调用格式比较简单，直接在程序中写出 continue 语句就可以了。下面使用一个简单的实例来说明 continue 命令的使用方法。

【例 5-29】通过简单的案例说明 continue 命令的使用方法。

（1）单击 MATLAB 命令窗口工具栏中的按钮，打开 M 文件编辑器。在 M 文件编辑器中输入以下程序代码。

```
break continue.m
for ii=1:9
     %   if ii==3       之所以在这段程序代码中的第二段代码前面添
         %   continue       加了%，是为了首先将其当作注释，不运行这段代码，
       %     end            在后面的程序中将注释符号删除后就可以重新运行。
        fprintf('ii=%d\n',ii);
          if ii==5
              break
          end
end
disp('The end of loop');
```

（2）将以上代码保存为“break_continue.m”文件，然后在 MATLAB 的命令窗口中输入 “break_continue”，按 Enter 键，可以得到对应的结果。

```
>> break continue
ii=1
ii=2
ii=4
ii=5
ii=6
ii=7
ii=8
ii=9
The end of loop
```

① 打开“break_continue.m”文件，然后在编辑器中修改其代码，得到的结果如下。

```
for ii=1:9
            if ii==3
                continue
           end
        fprintf('ii=%d\n',ii);
           if ii==5
               break
           end
end
disp('The end of loop');
```

② 在 MATLAB 的命令窗口中输入 “break_continue”，按 Enter 键，可以得到对应的结果。

```
>> break continue
ii=1
ii=2
ii=4
ii=5
The end of loop
```



在上面程序代码中使用了 break 语句，其功能就是跳出相应的程序代码。

❑ **终止循环——break 命令**

在 MATLAB 中，break 命令的功能在于终止本次循环，跳出最内层的循环，而不必等到循环的结束而是根据条件退出循环，常常和 if 语句结合运用来终止循环。

【例 5-30】 在 MATLAB 中寻求 Fibonacci 数组中第一个大于 700 的元素以及其数组标号。

（1）单击 MATLAB 命令窗口工具栏中的□按钮，打开 M 文件编辑器。在 M 文件编辑器中输入以下程序代码。

```
n=50;
     a=ones(1,n);
     for i=3:n
          a(i)=a(i-1)+a(i-2);
          if a(i)>=700
             a(i)
             break;
         end
     end
     i
```

（2）将以上代码保存为“Fib.m”文件，然后在 MATLAB 的命令窗口中输入“Fib”，按 Enter 键，就可以得到对应的结果。

```
>> Fib
ans =
   987
i =
16
```

从以上结果可以看出，在 Fibonacci 数组中第一个大于 700 的数值是 987，其对应的数组标号是 16。

❑ **转换控制——return 命令**

在通常情况下，当被调函数执行完后，MATLAB 会自动把控制转至主调函数或指定窗口。如果在被调函数中插入 return 命令，可以强制 MATLAB 结束执行该函数并把控制转出。

return 命令可以使正在运行的函数正常退出，并返回调用它的函数继续运行，经常用于函数的末尾，用来正常结束函数的运行。在 MATLAB 的内置函数中，很多函数的程序代码中引入了 return 命令，下面引用一个简要的 det 函数代码。

```
        function d = det(A)
        %DET det(A) is the determinant of A.
        if isempty(A)
             d=1;
            return
        else
            ...
        end
```

在以上程序代码中，首先通过函数语句来判断参数 A 的类型，当 A 是空数组时，直接返回 d=1，然后结束程序代码。

❑ **输入控制权——input 命令**

在 MATLAB 中，input 命令的功能是将 MATLAB 的控制权暂时交给用户，然后，用户通过键盘输入数值、字符串或表达式，按回车键将输入的内容输入到工作空间中，同时将控制权交还给 MATLAB，其常用的调用格式如下。

- **user_entry=input('prompt')** 将用户键入的内容赋给变量 user_entry。
- **user_entry=input('prompt.s')** 将用户键入的内容作为字符串赋给变量 user_entry。

对于以上第一个调用格式，可以输入数值、字符串、数组等各种形式的数据，第二个调用格式，无论用户输入怎样的变量，都会以字符串的形式赋给变量 user_entry。

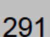
【例 5-31】 在 MAT LAB 中演示如何使用 input 函数。

（1）单击 MATLAB 命令窗口工具栏中的按钮，打开 M 文件编辑器。在 M 文件编辑器中输入以下程序代码。

```
    test input.m
        function test input()
        %在以上程序代码中，使用 isempty 来接收用户输入的“Enter”键，当什么字符
   %都不输入的时候，默认当用户输入的是 Y。
        reply = input('Do you want more?Y/N[Y]:','s');
        if isempty(reply)
            reply = 'Y';
        end
        if reply == 'Y'
            disp('you have selected more information');
        else
             disp('you have selected the end');
        end
```

（2）将以上代码保存为“test_input.m”文件，然后在 MATLAB 的命令窗口中输入“test_input”，然后按 Enter 键，就可以得到对应的结果。

```
      >> test input
      Do you want more?Y/N[Y]:
      you have selected more information
      >> test input
      Do you want more?Y/N[Y]:Y
      you have selected more information
      >> test_input
```

```
    Do you want more?Y/N[Y]:N
    you have selected the end
```

❑ 使用键盘——keyboard 命令

在 MATLAB 中，将 keyboard 命令放置到 M 文件中，将停止文件的执行并将控制权交给键盘。通过提示符 K 来显示一种特殊状态，只有当使用 return 命令结束输入后，控制权才交还给程序。在 M 文件中使用该命令，对程序的调试和在程序运行中修改变量都会十分便利。

【例 5-32】 在 MATLAB 中，演示如何使用 keyboard 命令。

（1）在 MATLAB 的命令窗口中输入以下内容。

```
>> keyboard
K>> for ii=1:9
    if ii == 3
        continue
    end
    fprintf('ii=%d\n',ii);
        if ii == 5
            break
        end
    end
ii=1
ii=2
ii=4
ii=5
K>> return
>>
```

（2）从以上程序代码可以看出，当输入 keyboard 命令后，在提示符的前面会显示 k 提示符，而当输入 return 后，提示符恢复正常的提示效果。



在 MATLAB 中，keyboard 命令和 input 命令的不同在于，keyboard 命令允许用户输入任意多个 MATLAB 命令，而 input 命令只能输入赋值给变量的数值。

❑ 提示警告信息——error 和 warning 命令

在 MATLAB 中，当编写 M 文件的时候经常需要提示一些警告信息。为此，MATLAB 提供了下面几个常见的命令（如表 5-4 所示）。

表 5-4　警告信息的命令列表

命　　令	说　　明
error('message')	显示出错信息 message，终止程序
errordlg('errorstring','dlgname')	显示出错信息的对话框，对话框的标题为 dlgname
warning('message')	显示警告信息 message，程序继续进行

【例 5-33】 使用不同的警告样式，查看 MATLAB 的不同错误提示模式。

（1）单击 MATLAB 命令窗口工具栏中的□按钮，打开 M 文件编辑器。在 M 文件编辑器中输入以下程序代码。

```
error_message.m
```

```
% Script file error message.m
%
% Purpose:
%  To calculate mean and the standard deviation of
% an input data set containing and arbitrary number
% of input values.
%
% Define variables:
% n       The number of input samples
% std dev   The standard devation of the input samples
% sum1     The sum of the input values
% sum2     The sum of the squares of the input values
% x       input data value
% xvar     The average of the input samples

% Initalize variables
sum1=0;sum2=0;

% Get the number of points to input
n=input('Enter the number of points: ');

% Check to see if we have enough input data.
if n<2
     errordlg('Not enough input data');
else
% construct if loop
for ii=1:n
    x=input('Enter value: ');
    sum1=sum1+x;
    sum2=sum2+x^2;
 end

%Calculate the mean and standard deviation
xvar=sum1/n;
std_dev=sqrt((n*sum2-sum1^2)/(n*(n-1)));

%Print the result
fprintf('The mean of this data set is: %f\n',xvar);
fprintf('The standard deviation is: %f\n',std_dev);
fprintf('The number of data is: %d\n',n);
end
```

（2）返回到 MATLAB 的命令窗口中，输入“error_message”，然后输入数值 1，得到的结果如图 5-8 所示。

当输入的数值总数小于 2 时，MATLAB 调用错误信息对话框。当单击对话框中的“OK”按钮后，将会自动退出程序代码。

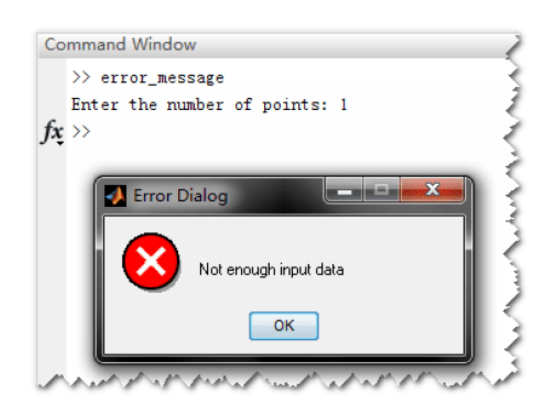


图 5-8　显示错误信息

（3）打开“error_message.m”文件，在编辑器中修改其程序代码，然后保存相应的程序代码，修改的程序代码如下。

```
sum1=0;sum2=0;
% Get the number of points to input
n=input('Enter the number of points: ');
% Check to see if we have enough input data.
if n<2
    error('Not enough input data');
else
    代码省略…
end
```

（4）返回到 MATLAB 的命令窗口中，输入“error_message”，然后输入数值 1，得到如下的结果。

```
>> error message
Enter the number of points: 1
??? Error using ==> error message at 24
Not enough input data
```

（5）打开“error_message.m”文件，在编辑器中修改其程序代码，然后保存相应的程序代码，修改的程序代码如下。

```
sum1=0;sum2=0;
% Get the number of points to input
n=input('Enter the number of points: ');
% Check to see if we have enough input data.
if n<2
    warning('Not enough input data');
else
    代码省略…
end
```

（6）返回到 MATLAB 的命令窗口中，输入“error_message”，然后输入数值 1，得到如下的结果。

```
>> error_message
```

```
Enter the number of points: 1
Warning: Not enough input data
> In error_message at 24
```

【例 5-34】 编写一个 M 函数文件，它具有以下功能：（1）根据指定的半径，画出蓝色圆周线；（2）可以通过输入字符串，改变圆周线的颜色、线型；（3）假若需要输出圆面积，则绘出圆。本例演示：M 函数文件的典型结构；指令 nargin,nargout 的使用和函数输入/输出量最大数目的柔性可变；switch-case 控制结构的应用示例；if-elseif-else 的应用示例；error 的使用。

（1）单击 MATLAB 命令窗口工具栏中的按钮，打开 M 文件编辑器。在 M 文件编辑器中输入以下程序代码。

```
sexangle.m
        function  [S,L]=sexangle(N,R,str)
        %sexangle.m          The area and perimeter of a regular polygon
    %   (正多边形面积和周长)
        %N                   The number of sides
        %R                   The circumradius
        %str        A line specification to determine line type/color
        %S                   The area of the regular polygon
        %L                   The perimeter the regular polygon
        % sexangle           用蓝实线画半径为 1 的圆
        % sexangle(N)            用蓝实线画外接半径为 1 的正 N 边形
        % sexangle(N,R)      用蓝实线画外接半径为 R 的正 N 边形
        % sexangle(N,R,str)  用 str 指定的线画外接半径为 R 的正 N 边形
        % S=sexangle(…)       给出多边形面积 S，并画相应正多边形填色图
        %[S,L]= sexangle(…) 给出多边形面积 S 和周长 L，并画相应正多边形填色图
        switch nargin
            case 0
                N=100;R=1;str='-b';
            case 1
                R=1;str='-b';
            case 2
                str='-b';
            case 3
             ;              %不进行任何变量操作，直接跳出 switch_case 控制结构
            otherwise
                error('输入量太多。');
        end;
        t=0:2*pi/N:2*pi;
        x=R*sin(t);
        y=R*cos(t);
        if nargout==0
            plot(x,y,str);
        elseif nargout>2
             error('输出量太多。');
        else
            S=N*R*R*sin(2*pi/N)/2;   %多边形面积
```

```
        L=2*N*R*sin(pi/N);        %多边形的周长
        fill(x,y,str)
    end
    axis equal square
    box on
    shg
```

（2）单击 M 文件编辑器中的“保存”按钮，将以上程序代码保存为“sexangle.m”。

（3）返回到 MATLAB 的命令窗口，根据输入不同的值，然后按 Enter 键，得到不同的图形。如“sexangle(6,2,'-g')”，如图 5-9 所示。“sexangle()”，如图 5-10 所示。“sexangle(6)”，如图 5-11 所示。“sexangle(6,2)”，如图 5-12 所示。

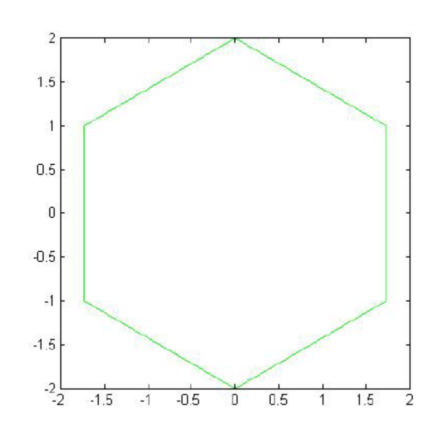


图 5-9　用绿色线画外接半径为 2 的正六边形

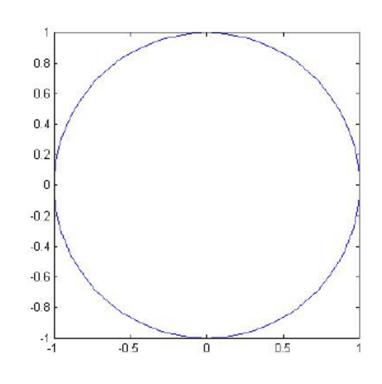


图 5-10　用蓝实线画半径为 1 的圆

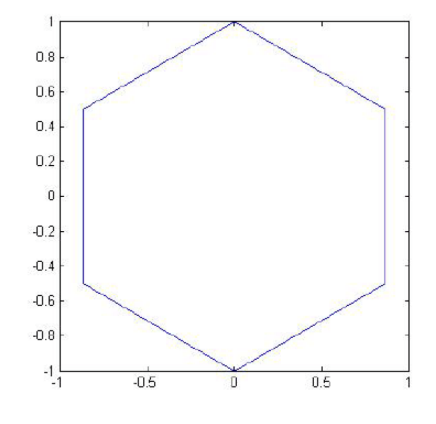


图 5-11　用蓝实线画外接半径为 1 的正六边形

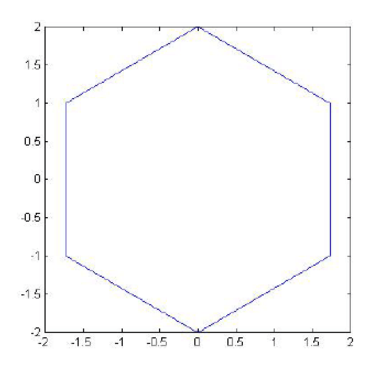


图 5-12　用蓝实线画外接半径为 2 的正六边形



函数定义名和保存文件名一致，两者不一致时，MATLAB 将忽视文件首行的函数定义名，而以保存文件名为准。

3．绘制抛物线轨迹——综合实例

对于比较复杂的 MATLAB 程序，经常需要将各种程序结构综合起来使用，才能解决复杂的问题。下面介绍一个比较简单的实例，来分析如何综合应用这些程序结构。

【例 5-35】 在 MATLAB 中，通过程序来演示小球的抛物线轨迹。

（1）分析小球的抛物线轨迹模型。假定用户抛小球的速度，也就是小球的初始速度是 v_0，小球的抛射初始角度是 θ。根据基础的物理知识可知，小球在水平和垂直方向上的速度分量分别为

$$\begin{cases} v_{x_0} = v_0 \cos\theta \\ v_{y_0} = v_0 \sin\theta \end{cases}$$

在本实例中，程序代码需要求解的是抛物线轨迹上水平距离的最长距离，根据相关知识，其距离的求解公式如下。

$$\begin{cases} t = -\dfrac{2v_{y_0}}{\mathrm{g}} \\ x_{\max} = v_{x_0} t \end{cases}$$

在以上公式中，g 代表的是重力加速度。在本实例中该参数选择的数值为-9.82，而对应的小球在垂直方向上的最高距离为

$$y_{\max} = \frac{v_{y_0}^2}{2\mathrm{g}}$$

（2）根据本实例的要求，可以输入抛射小球的初始速度，然后得出相应的计算数据。单击 MATLAB 命令窗口工具栏中的按钮，打开 M 文件编辑器，输入以下程序代码。

```
Ball.m
%Script file ball.m
%
%Purpose:
%  This program calculates the distance traveled by a ball
%throw at a speciified angle "theta" and a specified velocity
%"vo" from a point,ignoring air friction.It calculates the angle
%yeileding maximun range,and also plots selected trajectories.
%
%Define variable:
%conv      degrees to radians conv factor
%grav      The gravity accel
%ii,jj     Loop index
%index     The maximum range in array
%maxangle   The angle that gives the maximum range
%maxrange   Maximum range
%range     ranghe for a specified angle
%time      Time
%theta     Inital angle
%fly time   the totle trajectory time
%vo        The initial velocity
%vxo       x-component of the initial velocity
%vyo       y-component of the initial velocity
%x         x-position of ball
%y         y-position of ball
%定义常数数值
conv=pi/180;
grav=-9.82;
```

```
vo=input('Enter the initial velocity:');

range=zeros(1,91);
 %计算最大的水平距离
for ii=1:91
    theta =ii-1;
    vxo=vo*cos(theta*conv);
    vyo=vo*sin(theta*conv);
    max_time=-2*vyo/grav;
    range(ii)=vxo*max_time;
end
%显示计算水平距离的列表
fprintf('Range versus angle theta"\n');
for ii=1:5:91
    theta=ii-1;
    fprintf('%2d %8.4f\n',theta,range(ii));
end
%计算最大的角度和水平距离
[maxrange index]=max(range);
maxangle = index-1;
fprintf('\n Max range is %8.4f at %2d degress.\n',maxrange,maxangle);
%绘制轨迹图形
for ii=5:10:80
    theta=ii;
    vxo=vo*cos(theta*conv);
    vyo=vo*sin(theta*conv);
    max_time=-2*vyo/grav;
    %计算小球轨迹的 x,y 坐标数值
    x=zeros(1,21);
    y=zeros(1,21);
    for jj=1:21;
        time=(jj-1)*max_time/20;
        x(jj)=vxo*time;
        y(jj)=vyo*time+0.5*grav*time^2;
    end
    plot(x,y,'g');
    if ii==5
        hold on;
    end
end
    %添加图形的标题和坐标轴名称
    title('\bf Trajectory of Ball vs Inital Angle\theta');
    xlabel('\bf\itx \rm\bf(meters)');
    ylabel('\bf\ity \rm\bf(meters)');
    axis([0 max(range)+5 0 -vo^2/2/grav]);
    grid on;
    %绘制最大水平的轨迹图形
    vxo=vo*cos(maxangle*conv);
    vyo=vo*sin(maxangle*conv);
    max_time=-2*vyo/grav;
        %Calculate the (x,y)
```

```
        x=zeros(1,21);
        y=zeros(1,21);
        for jj=1:21
            time=(jj-1)*max_time/20;
            x(jj)=vxo*time;
            y(jj)=vyo*time+0.5*grav*time^2;
        end
        plot(x,y,'r','Linewidth',2);
        hold off
```

（3）单击 M 文件编辑器中的“保存”按钮，将以上程序代码保存为“ball.m”。

（4）返回到 MATLAB 的命令窗口，输入“ball”，然后按 Enter 键，根据程序代码的提示，依次输入不同的值，得到的结果如下。

```
>> ball
Enter the initial velocity:20
Range versus angle theta"
 0   0.0000
 5   7.0732
10  13.9316
15  20.3666
20  26.1828
25  31.2034
30  35.2760
35  38.2767
40  40.1144
45  40.7332
50  40.1144
55  38.2767
60  35.2760
65  31.2034
70  26.1828
75  20.3666
80  13.9316
85   7.0732
90   0.0000
Max range is  40.7332 at 45 degress.
```

除了以上数值结果之外，MATLAB 还会绘制相应的图形结果，如图 5-13 所示。

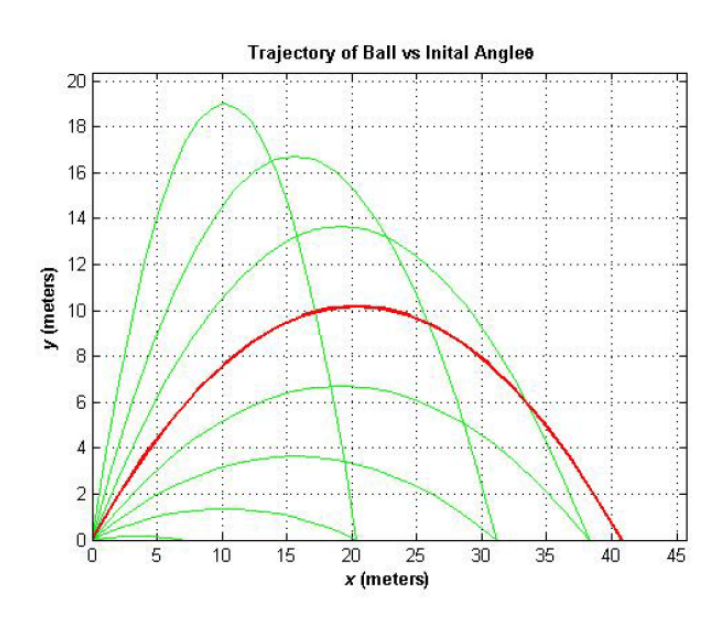
（5）修改初始速度数值，将其改为 45，得到的结果如下。

```
>> ball
Enter the initial velocity:45
Range versus angle theta"
 0   0.0000
 5  35.8083
10  70.5286
15 103.1059
20 132.5504
25 157.9674
30 178.5847
```



```
35 193.7757
40 203.0790
45 206.2118
50 203.0790
55 193.7757
60 178.5847
65 157.9674
70 132.5504
75 103.1059
80  70.5286
85  35.8083
90   0.0000
Max range is 206.2118 at 45 degress.
```

同时，MATLAB 会给出对应的图形结果，如图 5-14 所示。



5-13　初始速度为 2O 时的轨迹

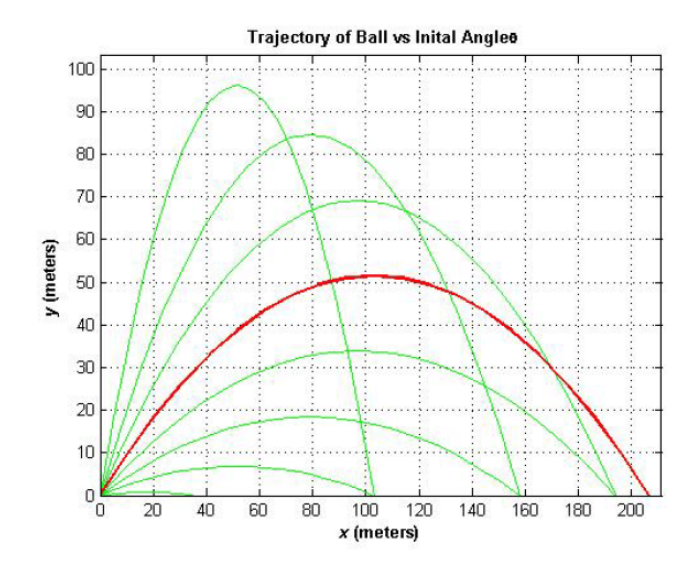


图 5-14　初始速度为 45 时的轨迹

5.4　脚本文件

M 文件可分为脚本文件（MATLAB scripts）和函数文件（MATLAB functions）。脚本文件是包含多条 MATLAB 命令的文件；函数文件可以包含输入变量，并把结果传送给输出变量。

脚本文件可以理解为简单的 M 文件，脚本文件中的变量都是全局变量。函数文件是在脚本文件的基础上添加了一行函数定义行，其代码组织结构和调用方式与对应的脚本文件截然不同。函数文件是以函数声明行“function...”作为开始的，其实质就是用户往 MATLAB 函数库里边添加了子函数，函数文件中的变量都是局部变量，除非使用了特别声明。函数运行完毕之后，其定义的变量将从工作区间中清除。而脚本文件只是将一系列相关的代码结合封装，没有输入参数和输出参数，即不自带参数，也不一定要返回结果。而多数函数文件一般都有输入和输出变量，并且有返回结果。

【例 5-36】 通过 M 脚本文件，画出下列分段函数所表示的画面。

$$p(x1,x2)=\begin{cases}0.5457e^{-0.75x_2{}^2-3.75x_1{}^2-1.5x_1} & x1+x2>1\\0.7575e^{-x_2{}^2-6x_1{}^2} & -1<x1+x2<=1\\0.5457e^{-0.75x_2{}^2-3.75x_1{}^2+1.5x_1} & x1+x2<=-1\end{cases}$$

Ex_5_36.m

```
a=2;
b=2;
clf;
x=-a:0.2:a;
y=-b:0.2:b;
for i=1:length(y)
    for j=1:length(x)
        if x(j)+y(i)>1
            z(i,j)=0.5457*exp(-0.75*y(i)^2-3.75*x(j)^2-1.5*x(j));
        elseif x(j)+y(j)<=1
            z(i,j)=0.5457*exp(-0.75*y(i)^2-3.75*x(j)^2+1.5*x(j));
        else z(i,j)=0.7575*exp(-y(i)^2-6*x(j)^2);
        end
    end
end
axis([-a,a,-b,b,min(min(z)),max(max(z))]);
colormap(flipud(winter));
surf(x,y,z);
```

将以上内容 M 文件 Ex_5_36.m 保存在当前目录下，然后在命令行输入该 M 文件的文件名，Ex_5_36，或者按下 F5 快捷键，或者打开文件后单击 M 文件编辑器的“运行”按钮，即可运行该文件，运行结果如图 5-15 所示。

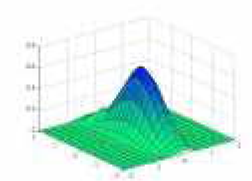


图 5-13 [illegible]

5.5 函数文件

MATLAB 中的函数主要有两种创建方法：在命令行中定义，保存为 M 文件。在命令行中创建的函数称为匿名函数，通过 M 文件创建的函数有多种类型，包括主函数、子函数及嵌套函数等。

5.5.1 主函数

主函数在结构上与其他函数没有区别，之所以称为主函数，是因为它在 M 文件中排在最前面，其他子函数都排在其后面。主函数与其 M 文件同名，是唯一可以在命令窗口或其他函数中调用的函数。主函数通过 M 文件名来调用。

本书前文涉及的函数文件都是主函数，所以这里就不再举例说明了。

5.5.2 子函数

一个 M 文件中可以写入多个函数定义式，排在第一个位置的是主函数，排在主函数后面进行定义的函数都叫子函数。子函数的排列无规定顺序。子函数只能被同一个文件上的主函数或其他子函数调用。子函数与主函数没有形式上的区别，每个子函数都有自己的函数定义行。

【例 5-37】 子函数示例。

```
newstats.m :
function [avg,med] = newstats(u)          % 主函数
% NEWSTATS Find mean and median with internal functions.
```

```
n=length(u);
avg = mean(u,n);
med = median(u,n);
function a = mean(v,n)                        %子函数
%Callculate average.
a=sum(v)/n;
function m = median(v,n)                       %子函数
%Callculate median
w = sort(v);
if rem(n,2) == 1
    m=w((n+1)/2);
else
    m=(w(n/2)+w(n/2+1))/2;
end
```

运行结果如下。

```
>> newstats 10
    ans =
             48.5000
```

本例中的主函数 newstats 用于返回输入变量的平均值和中位值，而子函数 mean 只是用来计算平均值，子函数 median 只是用来计算中位值，主函数在计算过程中调用了这两个子函数。需要注意的是，几个子函数虽然在同一个文件上，但各有自己的变量，子函数之间不能相互存取别人的变量。若声明变为全局变量，那另当别论。

1．调用一个子函数时的查找顺序

从一个 M 文件中调用时，MATLAB 首先查看被调用的函数是否是本 M 文件上的子函数，如果是，则调用它；如果不是，再寻找是否有同名的私有函数；如果还不是，则从搜索路径中查找其他 M 文件。因为最先查找的是子函数，所以在 M 文件中可以编写子函数来覆盖原有的其他同名函数文件。例如【例 5-37】中子函数名称 mean 和 median 是 MATLAB 内建函数，但是通过子函数的定义，可以调用自定义的 mean 和 median 函数。

2．子函数的帮助文本

可以像为主函数写帮助文本那样为子函数写帮助文本。但是，显示子函数的帮助文本有点区别，要把 M 文件名加在子函数名前面。如子函数名为 mysubfun，放在 myfun.m 文件上。要在命令行得到它的帮助信息，需输入命令。

```
help myfun>mysubfun(“>”之前之后不能有空格)
```

【例 5-38】 子函数的帮助文本查看示例。

```
>> help newsstats>mean
 Callculate average.
>> help mean
 MEAN   Average or mean value.
    For vectors, MEAN(X) is the mean value of the elements in X. For
    matrices, MEAN(X) is a row vector containing the mean value of
    each column.  For N-D arrays, MEAN(X) is the mean value of the
    elements along the first non-singleton dimension of X.
```

```
MEAN(X,DIM) takes the mean along the dimension DIM of X.

Example: If X = [1 2 3; 3 3 6; 4 6 8; 4 7 7];

then mean(X,1) is [3.0000 4.5000 6.0000] and
mean(X,2) is [2.0000 4.0000 6.0000 6.0000].'

Class support for input X:
   float: double, single

See also median, std, min, max, var, cov, mode.

Overloaded methods:
   timeseries/mean
   fints/mean
   ProbDistUnivParam/mean

Reference page in Help browser
   doc mean
```

5.5.3 私有函数

私有函数实际上是另一种子函数，它是私有的，只有父 M 文件函数能调用它。私有函数的存储需要在当前目录下建一个子目录，子目录名字必须为 private。存放于 private 文件夹内的函数即为私有函数，它的上层目录称为父目录，只有父目录中的 M 文件才可以调用私有函数。

❑ 私有函数对于其父目录以外的目录中的 M 文件来说是不可见的。

❑ 调用私有函数的 M 文件必须在 private 子目录的直接父目录内。

假如私有函数名为 myprivfun，为了得到私有函数的帮助信息，需输入如下命令。

```
help private/myprivfun
```

私有函数只能被其父文件夹中的函数调用，因此，用户可以开发自己的函数库，函数名称可以与系统标准 M 函数库名称相同，而不必担心在函数调用时发生冲突，因为 MATLAB 首先查找私有函数，然后查找标准函数。

5.5.4 嵌套函数

所谓嵌套函数，是指在某函数中定义的函数。

1．写嵌套函数

MATLAB 允许在函数 M 文件的函数体中，定义一个或多个嵌套函数。像任何 M 文件函数一样，被嵌套的函数能包含任何构成 M 文件的成分。

MATLAB 函数文件一般不需要使用 end 语句来表征函数体已经结束。但是嵌套函数，无论是嵌套的还是被嵌套的，都需要以 end 语句结束。而且在一个 M 文件内，只要定义了

嵌套函数，其他非嵌套函数也要以 end 语句结束。

最简单的嵌套函数的结构如下。

```
function x = A(p1,p2)
…
    function y=B(p3)
    …
    end
…
end
```

另外，一个主函数还可以嵌套多个函数，例如，多个平行嵌套函数结构如下。

```
function x = A(p1,p2)
…
    function y=B(p3)
    …
    end

    function z=C(p4)
    …
    end
…
end
```

在这个程序中，函数 A 嵌套了函数 B 和 C，嵌套函数 B 和 C 是并列关系。除了平行嵌套函数外，还有多层嵌套函数。

```
function x = A(p1,p2)
…
    function y=B(p3)
    …
        function z=C(p4)
        …
        end
    …
    end
…
end
```

在这段程序中，函数 A 嵌套了函数 B，而函数 B 嵌套了函数 C。

2．嵌套函数的调用

一个嵌套函数可以被下列函数调用。

① 该嵌套函数的直接上一层函数；

② 同一母函数下的同级嵌套函数；

③ 被任一低级别的函数调用。

【例 5-39】 嵌套函数调用示例。

Ex_5_39.m



```
function A(x,y)                    %主函数
B(x,y);
D(y);
    function B(x,y)                %嵌套在 A 内
        C(x);
        D(y);
        function C(x)              %嵌套在 B 内
            D(x);
        end
    end
    function D(x)                  %嵌套在 A 内
        E(x);
        function E(x)              %嵌套在 D 内
            ...
        end
    end
end
```

在这段程序中，函数 A 包含了嵌套函数 B 和嵌套函数 D。函数 B 和函数 D 分别嵌套了函数 C 和函数 E。这段程序中函数间的调用关系如下。

- 函数 A 为主函数，可以调用函数 B 和函数 D，但是不能调用 C 和函数 E。
- 函数 B 和函数 D 为同一级嵌套函数，B 可以调用 D 和 C，但是不能调用 E; D 可以调用 B 和 E，但是不能调用 C。
- 函数 C 和函数 E 为分属两个函数的嵌套函数，C 和 E 都可以调用 B 和 D;虽然它们属于同级别的函数，但是分属于不同的母函数，所以不能互相调用。

3. 嵌套函数中变量的使用范围

通常在函数之间，局部变量是不能共享的。子函数不能与主函数或其他子函数共享变量，因此，每个函数都有自己的工作空间（workspace），用于存放自己的变量。嵌套函数也都有自己的工作空间。但是因为它们是嵌套关系，所以有些情况下可以共享变量。

【例 5-40】 嵌套函数示例 1。

```
varScope1.m
function varScope1
x = 5;
nestfun1
        function nestfun1
            nestfun2
            function nestfun2
                x = x+1
            end
        end
end
varScope2.m
function varScope2
nestfun1
        function nestfun1
```

```
            nestfun2
            function nestfun2
                x=5;
            end
        end
x=x+1
end
```

运行结果如下。

```
>> varScope1
x =
     6
>> varScope2
x =
     6
```

本例中的两个 M 文件都使用了多层嵌套函数。在这两个例子中，变量 x 被储存在外层主函数的工作空间，所以它可被嵌套在里面的函数读取或写入。

【例 5-41】 嵌套函数示例 2。

```
varScope3.m
function varScope3
nestfun1
nestfun2
       function nestfun1
           x=5;
       end
       function nestfun2
           x=x+1
       end
end
```

本例中的两个嵌套函数 nestfun1 和 nestfun2 是并列关系，外层的函数 varScope3 没有读取 x，因为 x 不在其工作空间中，所以，x 并不能被两个嵌套函数共享。nestfun1 定义了 x 在 nestfun1 的工作空间中，不能被 nestfun2 共享。因此，当 nestfun2 运行之后试图访问 x 时，就会出错。运行本例中的程序，将会显示如下错误信息。

```
>> varScope3
??? Undefined function or variable "x".

Error in ==> varScope3>nestfun2 at 8
       x=x+1

Error in ==> varScope3 at 3
nestfun2
```

【例 5-42】 嵌套函数输出变量的共享示例。

```
varScope4.m
function varScope4
x=5;
nestfun;
      function y = nestfun
         y=x+1;
      end
y
end
varScope5.m
function varScope5
x=5;
z=nestfun;
      function y = nestfun
         y=x+1;
      end
z
end
```

由嵌套函数返回的结果变量并不被外层的函数共享。在 varSeope4.m 和 varScope5.m 中，varScopc4.m 在运行到倒数第 2 行时会发生错误。这是因为虽然在嵌套函数中计算并返回了 y 的值，但是这个变量 y 只存在于嵌套函数的工作空间，并不能被外层函数共享。而在 varScape5.m 中将嵌套函数赋值给了变量 z，所以最终可以正确地显示 z 的值，具体的运行结果如下。

```
>> varScope4
??? Undefined function or variable 'y'.
Error in ==> varScope4 at 7
y
>> varScope5
z =
    6
```

5.5.5 重载函数

重载函数是已经存在的函数的另外版本。假设有一个函数是为某种特定的数据类型设计的，当要使用另外类型的数据时，就要重写此函数，使其能处理新的数据类型，但它的名字与原函数名相同。至于调用函数的哪个版本，则取决于数据类型和参数的个数。

每个重载的 MATLAB 函数，都有一个 M 文件放在 MATLAB 目录中。同一种数据类型的不同重载函数 M 文件放在同一个目录下。目录以这种数据类型命名，并用@符号开头。例如，在目录\@ double 下的函数，在输入变量数据类型为 double 时才可以被调用；而在目录\@int32 下的函数，则在输入变量数据类型为 int32 时才可以被调用。

5.6 P 码文件和变量使用范围

下面补充介绍两个重要的概念，一个是 P 码文件，一个是变量的使用范围。

5.6.1 P 码文件

1. 语法分析过程和伪代码

一个 M 文件首次被调用（运行文件名，或被 M 文本编辑器打开）时，MATLAB 将首先对该 M 文件进行语法分析（Parse），并把生成的相应内部伪代码（Psedecode，简称 P 码）文件存放在内存中。此后，当再次调用该 M 文件时，将直接调用该文件在内存中的 P 码文件，而不会对原码文件重复进行语法分析。值得注意的是，MATLAB 的分析器（Parse）总是把 M 文件连同被它调用的所有函数 M 文件一起变换成 P 码文件。

P 码文件有与原码文件相同的文件名，但其扩展名是“.p”。本质上说，P 码文件运行速度高于原码文件。

在 MATLAB 中，假如存在同名的 P 码和原码文件，那么，当该文件名被调用时，被执行的肯定是 P 码文件。

2. P 码文件的预生成

P 码文件不是仅当 M 文件被调用时才可产生，P 码文件也可被预先生成，具体如下。

- **pcode FunName**　在当前目录上生成 FunName.p。
- **pcode FunName –inplace**　在 FunName.m 所在目录上生成 FunName.p。

3. 内存中 P 码文件的列表和清除

- **inmem**　罗列出内存中所有 P 码文件名。
- **clear FunName**　清除内存中的 FunName.pP 码文件。
- **clear functions**　清除内存中的所有 P 码文件。



P 码文件较之原码文件有两大优点——是运行速度快，对于规模较大的问题其效果尤为显著；二是由于 P 码文件是二进制文件，难于阅读，因此，用户常借助其为自己的程序保密。

【例 5-43】 在 MATLAB 中，查看内存中的所有 P 码文件，然后清除所有 P 码文件，再次查找内存中的 P 码文件信息。

在 MATLAB 的命令窗口中输入以下代码。

```
>> inmem
ans =
'matlabrc'
'pathdef'
'userpath'
…
'codetools\private\dataviewerhelper'
'path'
'mdbstatus'
'breakpointsForAllFiles'
'+editor\private\createJavaBreakpointsFromDbstatus'
…
'ismember'
'iscellstr'
'unique'
```

```
'sortrows'

>> clear functions
>> inmem

ans =
Empty cell array: 0-by-1
```

5.6.2 局部变量、全局变量和持存变量

MATLAB 有共三种类型的变量：局域、全域和持存变量。

1. 局域变量

对 M 函数文件（除内嵌函数）而言，除该函数的输入、输出量外，每个函数文件中所用到的变量，都是局域变量（Local Variables）。

- 局域变量的作用域（Variable Scope）仅限于该函数本身，它存放于隶属该函数的专用内存空间。各函数的内存空间是相互独立的、互不相通的口。
- 局域变量仅生存于该函数的运行过程期间。一旦函数运行结束，该函数内存空间连同其中保存的变量就全被清空并释放。

2. 全域变量

通过 global 指令。MATLAB 也允许几个不同的函数空间及基本工作空间共享同一个变量，这种被共享的变量称为全域变量（Global Variables）。

（1）全域变量必须专门特别声明

- 每个希望共享全域变量(比如名为 DELTA 的变量)的函数或 MATLAB 基本工作空间必须各自用 global 指令对具体变量加以声明。没采用 global 指声明的函数或基本工作空间，将无权享用全局变量。例如，把 DELTA 声明为全域变量的格式为 global DELTA
- 对具体变量的“全域化”声明，必须在每个函数的其他指令运行前进行。

（2）全域变量将影响与之关联的所有内存空间

- 如果某个函数的运作使全域变量的内容发生了变化，那么，其他函数空间以及基本工作空间中的同名变量也就随之变化。
- 全域变量是否存在不受与它关联的函数运行与否的影响。只有当该全域变量关联的全部函数被清除，并同时把该全域变量从基本内存空间删除的情况下，全域变量才消失。Clear all 可以执行这个删除功能。

（3）全域变量应用旨要

- 由于全域变量损害函数的封装性，因此，应尽量避免使用全域变量，以免出现难以觉察的程序失误。在可能的情况下，尽量使用持存变量代替全域变量。
- 由于全域变量关联面广，其变量名建议尽量用“大写字符”及“较多字符”组成，以免不经意间的错用。
- 可以使用指令 whos global，检查内存空间中是否存在“全域变量”。

3. 持存变量

通过 persistent 指令，MATLAB 也允许几个不同的函数空间共享同一个变量，这种在函数间被共享的变量称为持存变量（Persistent Variables）。

（1）持存变量必须特别声明

- ❑ 每个希望共享持存变量(比如名为 Sigma 的变量)的函数，必须在各自函数体内用 persistent 指令对具体变量加以声明。没采用 persistent 指令声明的函数或基本工作空间，将无权享用全局变量，例如，把 Sigma 声明为持存变量的格式为 persist Sigma
- ❑ 对具体变量的“全域化”声明，最好在每个函数的其他指令运行前进行。

（2）持存变量将影响与之关联的所有内存空间

- ❑ 如果某个函数的运作使持存变量的内容发生了变化，那么，其他函数空间中的同名变量也就随之变化。
- ❑ 持存变量是否存在不受与它关联的函数运行与否而影响，只有当与该持存变量关联的全部函数被删除的情况下，持存变量才消失。

（3）持存变量与全域变量的区别

持存变量应用于函数与基（内存）空间无关；而全域变量跟函数及基空间都有关。

5.7 M 文件调试

用户在编写 M 文件程序时会有错误在所难免，能够熟练掌握调试的方法和技巧可以提高工作效率。

5.7.1 M 文件出错信息

在创建 M 文件过程中，会遇到两类错误：语法（Syntax）错误和运行（Run-time）错误。M 文件编辑器的语法错误检测功能，已经在前两节进行了描述。本节将集中介绍发现和纠正运行错误的调试（Debugging）方法和辅助工具。

运行错误发生在程序执行过程中。相对语法错误而言，动态的运行错误较难发现和处理，其原因在于如下几点。

- ❑ 运行错误来源于算法模型与期望目标是否一致；程序模型与算法是否一致，这涉及用户对期望目标原理的理解、对算法的理解，还涉及用户对 MATLAB 指令的理解、对程序流的理解和对 MATLAB 工作机理的理解。
- ❑ 运行错误的表现形态较多。如程序正常运行，但结果错误；程序不能正常运行而中断。
- ❑ 运行错误是动态错误。尤其是 M 函数文件，它一旦运行停止，其中间变量被删除一空，错误查找很难着手。

5.7.2 M 文件调试方法

本节将介绍两种调试（Debug）方法：直接调试法和工具调试法。

1. 直接调试法

由于 MATLAB 语言本身的向量化程度高，程序一般都显得相对简单，再加上 MATLAB 语言的可读性强，因此，直接调试法往往十分奏效。直接调试法包括以下一些手段。

- 将重点怀疑语句行、指令行后的分号“;”删除或改成“,”，使计算结果显示于屏幕。
- 在适当的位置，添加显示某些关键变量值的语句(包括使用 disp 在内)。
- 利用 echo 指令，使运行时，在屏幕上逐行显示文件内容。echo on 能显示 M 脚本文件；echo FunName on 能显示名为 FunName 的 M 函数文件。
- 在原 M 脚本或函数文件中的适当位置，增添 keyboard 指令。当 MATLAB 运行至 keyboard 指令时，将暂停执行文件，并在 MATLAB 指令窗中出现 K 提示符，此时，用户可以输入指令查看基本内存空间或函数内存空间中存放的各种变量，也可以输入指令修改那些变量。在 k 提示符后键入 raturn 指令，结束查看，原文件继续往下执行。
- 通过在原函数文件首行之前加上百分号，使一个中间变最难于观察的 M 两数文件变为一个所有变量都保留在基空间中的 M 脚本文件。

如果函数文件规模很大，文件内嵌套复杂，有较多的函数、子函数、私用函数调用，直接调试法可能失败，那么，可借助 MATLAB 提供的专门工具——调试器（Debugger）进行。

【例 5-44】 在 MATLAB 中，使用直接调试法来调试程序代码。

（1）单击 MATLAB 命令窗口工具栏中的按钮，打开 M 文件编辑器，输入以下程序代码。

```
    ballw. m
function f=ballw(K,ki)
% ballw.m 演示红色小球沿一条封闭螺线运动的实时动画
% 仅演示实时动画的调用格式为 ballw(K)
% 既演示实时动画又拍摄照片的调用格式为 f=ballw(K,ki)
% K 红球运动的循环数(不小于 1 )
% ki 指定拍摄照片的瞬间，取 1 到 1034 间的任意整数。
% f 存储拍摄的照片数据，可用 image(f.cdata) 观察照片。
% 产生封闭的运动轨线
t1=(0:1000)/1000*10*pi;x1=cos(t1);y1=sin(t1);z1=-t1;
t2=(0:10)/10;x2=x1(end)*(1-t2);y2=y1(end)*(1-t2);z2=z1(end)*ones(size(x2
));
t3=t2;z3=(1-t3)*z1(end);x3=zeros(size(z3));y3=x3;
t4=t2;x4=t4;y4=zeros(size(x4));z4=y4;
x=[x1 x2 x3 x4];y=[y1 y2 y3 y4];z=[z1 z2 z3 z4];
%data=[x',y',z']
plot3(x,y,z, 'y','Linewidth',2 ), axis off % 绘制曲线
% 定义 " 线 " 色、 " 点 " 型(点)、点的大小( 40 )、擦除方式( xor)
h=line( 'Color' ,[0.67 0 1], 'Marker' , '.' , 'MarkerSize' ,40, 'EraseMode' ,
'xor' );
% 使小球运动
n=length(x);i=1;j=1;
while 1 % 无穷循环
set(h, 'xdata' ,x(i), 'ydata' ,y(i), 'zdata' ,z(i));
```

```
%bw=[x(i),y(i),z(i)]% 小球位置
drawnow; % 刷新屏幕 <21>
pause(0.0005) % 控制球速 <22>
i=i+1;
if nargin==2 & nargout==1 % 仅当输入宗量为 2 、输出宗量为 1 时，才拍摄照片
if (i==ki&j==1);f=getframe(gcf); end % 拍摄 i=ki 时的照片 <25>
end
if i>n
i=1;j=j+1;
if j>K; break ; end
end
end
```

（2）将以上程序代码保存为“ballw. m”文件，然后在命令窗口中输入“ballw(2,200)”，得到的图形如图 5-16 所示。当 MATLAB 完成了以上程序代码后，得到的结果如图 5-17 所示。

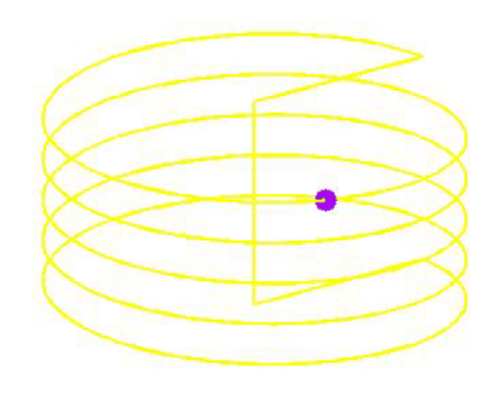


图 5-16　程序运行的结果

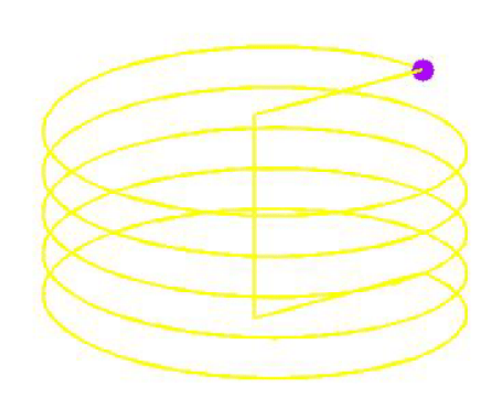


图 5-17　程序的最终结果图形

（3）显示封闭曲线的坐标数值。打开保存的“ballw.m”文件，然后将程序代码修改如下。

```
…
t3=t2;z3=(1-t3)*z1(end);x3=zeros(size(z3));y3=x3;
t4=t2;x4=t4;y4=zeros(size(x4));z4=y4;
x=[x1 x2 x3 x4];y=[y1 y2 y3 y4];z=[z1 z2 z3 z4];
data=[x',y',z']            %添加显示封闭曲线的坐标数值(此为修改部分)
plot3(x,y,z, 'y','Linewidth',2 ), axis off % 绘制曲线
% 定义 " 线 " 色、 " 点 " 型(点)、点的大小( 40 )、擦除方式( xor)
…
```

（4）查看程序结果。返回到命令窗口，输入命令行“ballw(2,200)”，得到的结果如下。

```
        >> ballw(2,200)

data =

    1.0000         0         0
    0.9995    0.0314   -0.0314
    0.9980    0.0628   -0.0628
```

```
    0.9956    0.0941   -0.0942
    0.9921    0.1253   -0.1257
    0.9877    0.1564   -0.1571
    0.9823    0.1874   -0.1885
    …  //限于篇幅，省略了部分数据
         0         0  -15.7080
         0         0  -12.5664
         0         0   -9.4248
         0         0   -6.2832
         0         0   -3.1416
         0         0         0
         0         0         0
    0.1000         0         0
    0.2000         0         0
    0.3000         0         0
    0.4000         0         0
    0.5000         0         0
    0.6000         0         0
    0.7000         0         0
    0.8000         0         0
    0.9000         0         0
    1.0000         0         0
```

从以上程序结果可以看出，当在程序代码中添加一个简单的语句“data= [x',y',z']”后，就可以在程序代码执行的过程中，查看封闭曲线的所有坐标值数值。如果程序结果中封闭曲线不正常，则可以从以上数据中查看效值的问题。

（5）显示小球位置的坐标数值。打开上面步骤保存的“ballw.m”文件，然后将程序代码修改。

```
while 1 % 无穷循环
set(h, 'xdata' ,x(i), 'ydata' ,y(i), 'zdata' ,z(i));
bw=[x(i),y(i),z(i)]% 小球位置(此为修改部分)
drawnow; % 刷新屏幕
```

（6）查看程序结果。返回到命令窗口，输入命令行“ballw(2,200)”，得到的结果如下。

```
bw =
     1     0     0
bw =
    0.9995    0.0314   -0.0314
bw =
    0.9980    0.0628   -0.0628
bw =
    0.9956    0.0941   -0.0942
bw =
    0.9921    0.1253   -0.1257
…
```

在以上步骤中，分别使用简单程序代码查看出关键的程序数据，如果在程序运算过程中出现问题，则可以从以上程序数值中查看出相应的问题。

2．工具调试法

MATLAB 不但向用户提供了专门的指令式调试工具，而且在 M 文件编辑器上集成有图示式调试装置（Graphical Debugger）。如图 5-18 所示展示了一组调试图标、设置的断点和程序暂停指针等。表 5-5 列出了各调试图标的功用。

表 5-5　调试功能键、菜单选项和相应指令对照表

功能键	含　　义	相应的菜单条选项	相应指令行指令
	断点设置（或清除）	{Breakpoints; Set/Clear Breakpoint}	Dbstop/ dbclear
	清除全部断点	{Breakpoints; Clear All Breakpoints }	Dbclear all
	单步执行	{Debug;Stop}	dbstep
	进入被调函数	{Debug;Step In}	dbstep in
	跳出被调函数	{Debug;Step Out}	dbstep out
	连续执行	{Debug;Continue}	dbcont
	结束调用	{Debug;Exit Debug Mode}	dbquit

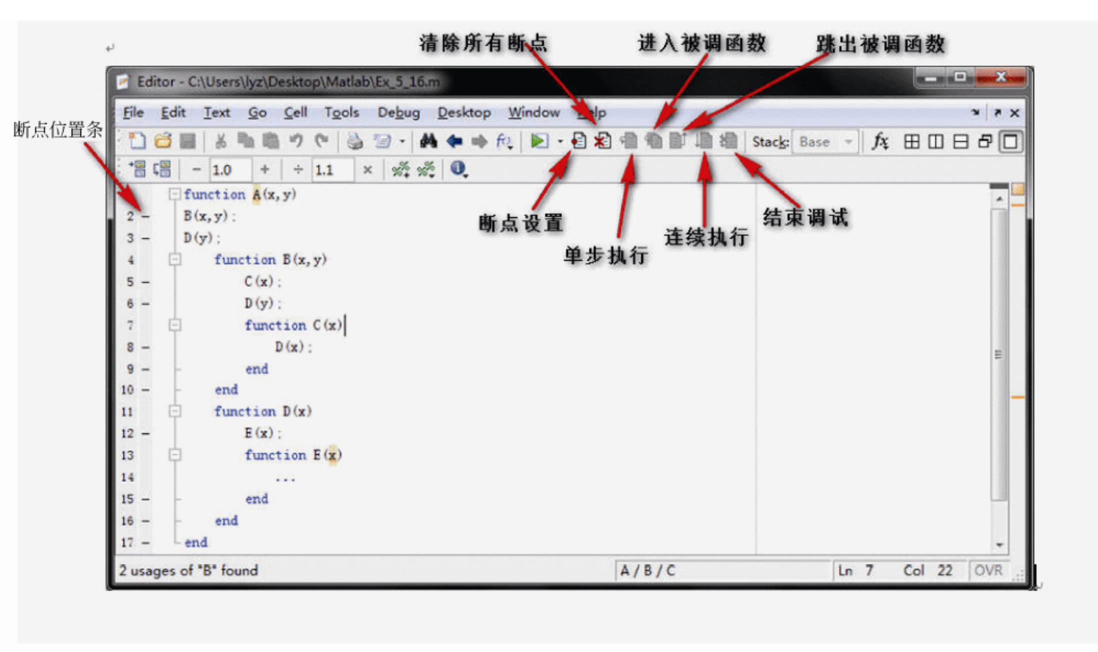


图 5-18　M 文件编辑/调试器

M 文件编辑/调试器的编辑功能在第 1 节已经阐述，本节集中介绍调试器功能与使用方法。



（1）设置断点的两种方法。

- 直接单击法(推荐使用)　在调试器界面的“断点位置条”中，单击“所需中断行”左侧的“短线条”，就会出现“红色断点标志”。
- 工具图标法　把光标置于“所需中断行”，然后单击工具图标，于是该行的左侧“短线条”就变成“红色断点标志”。

（2）撤销断点的两种方法。

- ❑ 直接单击法(推荐使用)　用鼠标单击“所需撤销的红色断点标志”，该红点就变回“短线条”，于是该断点被撤销。
- ❑ 工具图标法　把光标置于“所需撤销断点的行”，然后单击工具图标，于是那行的左侧“红色断点标志”变回“短线条”，断点被撤销。

（3）程序执行指针

- ❑ 程序进行调试状态后，在调试器中就会出现标志程序进程的“绿色的指针➡”。
- ❑ 在整个调试过程中，“绿色的指针➡”随各种(如单步、进入、跳出等)调试操作而运动，它醒目地展示了程序的进程。

1．调试器应用示例

正如前面所说，由于 M 文件错误的多样性，调试器的具体使用方法会随具体问题而变化，下面通过实例叙述调试器的基本使用方法。

【例 5-45】 本例的目标：对于任意随机向量，画出鲜明标志该随机向量均值、标准差的频数直方（如图 5-19 所示），或给出绘制这种图形的数据。

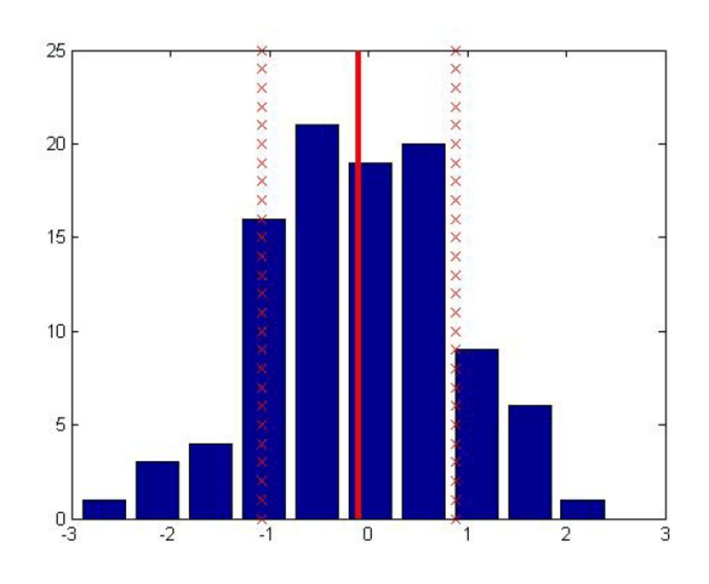


图 5-19 带均值、标准差标志的频数直方图

（1）根据题目要求写出以下两个 M 文件。

```
barzzy1.m
function [nn,xx,xmu,xstd]=barzzy1(x)
%本函数文件专供实践调试器用
xmu=mean(x);
xstd=std(x);
[nn,xx]=hist(x);
if nargout==0
    barzzy2(nn,xx,xmu,xstd)                                %<7>
end
barzzy2.m
```

```
    function barzzy2(nn,xx,xmu,xstd)
%本函数供 barzzy1.m 调用
%本函数故意设置了一个错误
clf,
bar(xx,nn);hold on
Ylimit=get(gca,'YLim');
yy=0:Ylimit(2);
xxmu=xmu*size(yy);
xxL=xxmu/xmu*(xmu-xstd);
xxR=xxmu/xmu*(xmu+xstd);
plot(xxmu,yy,'r','Linewidth',3)                    %<11>
plot(xxL,yy,'rx','MarkerSize',8)
plot(xxR,yy,'rx','MarkerSize',8),hold off
```

（2）初次运行以下指令后，得到运行出错的提示，如图 5-20 所示。

```
K>> randn('seed',1),x=randn(1,100);
barzzy1(x);
??? Error using ==> plot
Vectors must be the same lengths.

Error in ==> barzzy2 at 11
plot(xxmu,yy,'r','Linewidth',3)                      %<11>

Error in ==> barzzy1 at 7
    barzzy2(nn,xx,xmu,xstd)                          %<7>
```

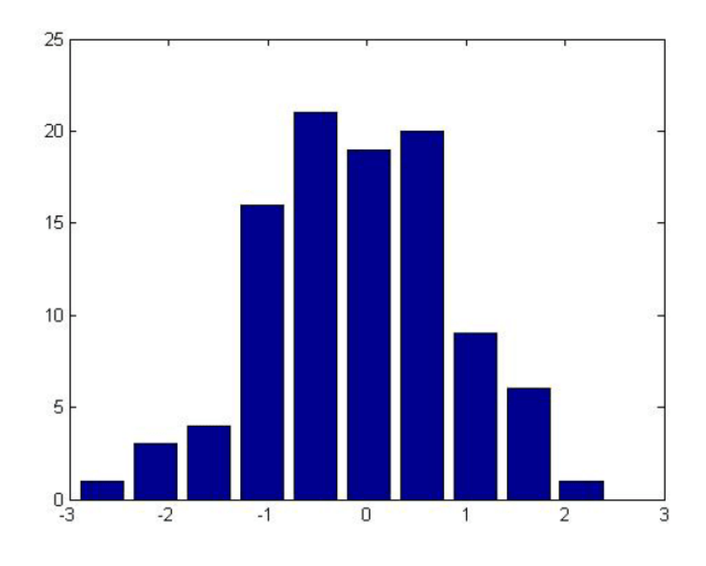


图 5-20　运行出错时所得的不完整图形

（3）初步分析错误原因。

根据提示可知，问题发生在 barzzy2.m 文件 plot 指令中的 xxmu 和 yy 两个文件的向量的长度不同。于是要查这两个向量到底是什么？长度不同的根源在何处?

由于错误发生在函数 barzzy2.m 中，所以在错误发生后，该函数空间中变量都全部消失。为此，使用调试器进行调试。

（4）断点设置。

操作方法：单击 barzzy1.m 第 6 行“断点位置条”中的“短线条”，出现断点标注●（红点）。在 barzzy2.m 函数的第 9 行，进行类似的操作，实现断点设置。

（5）进入调试状态。

在指令窗中运行以下指令，就进入动态调试。

```
randn('seed',1),x=randn(1,100); barzzy1(x);
```

该指令的运行，引起两个窗口发生如下变化。

① 指令窗出现“控制权交给键盘”的标志符 K>>，如图 5-21 所示。

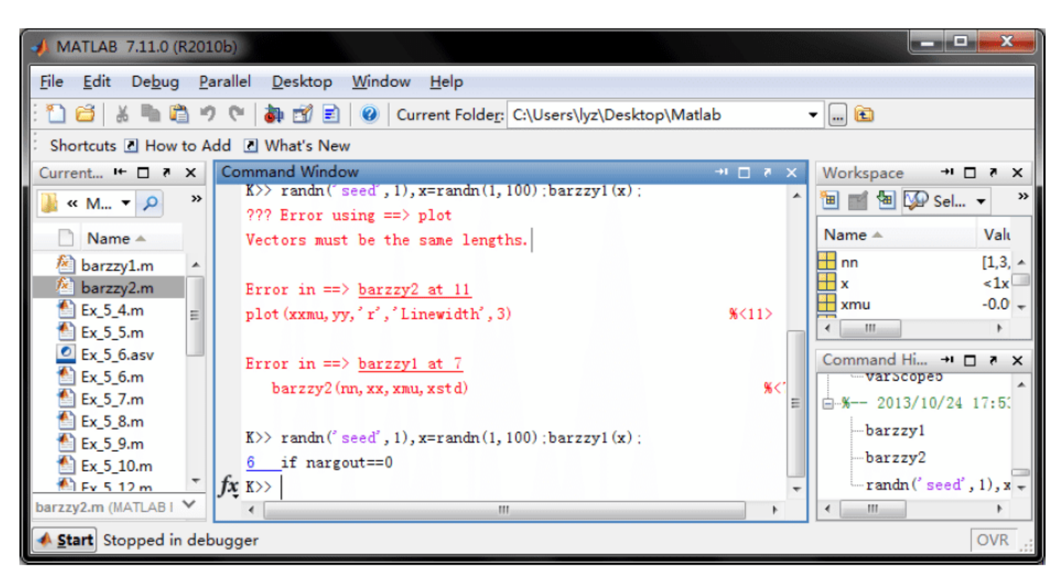


图 5-21　进入调试状态的指令窗

② barzzy1.m 所在编辑/调试器窗口中的变化如下。

❑ 在所设的第一断点旁出现“绿色右指箭头●➡”，该调试指针表明运行中断在此行之前。

❑ 编辑/调试器右上方的“内存菜单”栏显示“barzzy1.m”字样，表示目前处在函数内存空间中。

（6）进入被调文件 barzzy2.m 函数内部。

单击工具条上的“进入被调的函数”图标心，引出 barzzy2.m 文件的调试窗口，不管原先 barzzy1.m.文件是否已经被打开，只要该文件在搜索路径上，调试指针停留在函数文件可执行指令的首行。

（7）连续执行，直到另一个断点。

单击“连续执行”功能键，就使程序执行完第 8 行指令后，停止在第 9 行指令。

（8）观察这段程序运行后产生的中间结果，确定错误的准确位置。

❑ 观察指令 plot 中的 yy 变量。

观察运行所生成变量的常用方法有下列三种。

① 变量值的鼠标观察法(可快捷观察较小规模变量值)。把鼠标移到待观察变量处，就可看到变量内容。如图 5-22 所示，鼠标放在 yy 变量名上，就看到 yy 是长度为 26 的向量。

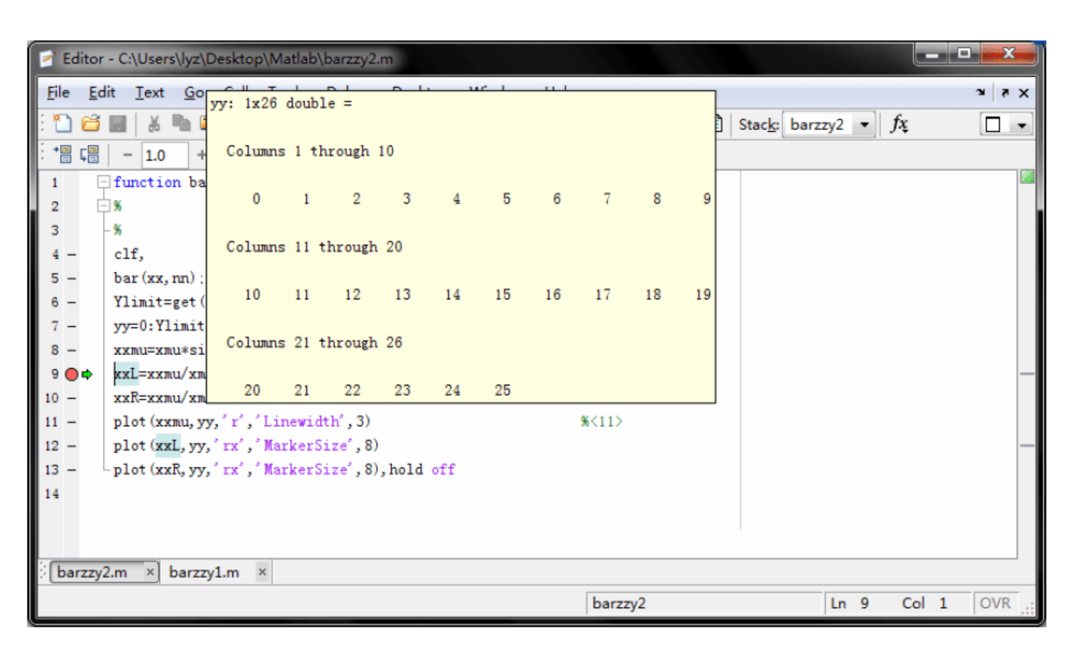


图 5-22　变量值的鼠标观察法

② 指令窗观察法（适于观察较大规模变量值）。在 K 提示符后，键入变量名，就会显示出相应的变量值。

③ 变量编辑器观察法（适于观察大规模变量值）。此时，MATLAB 操作桌面上的“工作空间浏览器”中，展现 barzzy2.m 函数内存空间中的所有变量；双击希望观察的变量，就能在“变量编辑器”中看到变量值。

- ❑ 观察第 9 行指令 plot 中的另一个变量 xxmu，发现它仅是长度为 2 的向量。显然，错误是由 xxmu 和 yy 两个向量长度不一致引起的。
- ❑ 由第 9 行指令向上追溯，又可以发现，这错误源于第 6 行指令。

编写该行指令的原意是产生一根与 yy 长度相同的 xxmu 向量，以便用于绘制一条垂直横轴的直线。但是，该行指令写错了，正确写法应是 xxmu=xmu*ones(size(yy))。

（9）修改程序，停止第一轮调试，重新运行。

- ❑ 单击“结束调试”功能键。
- ❑ 把 barzzy2.m 文件第 8 行指令改写为 xxmu=xmu*ones(size(yy))，并进行文件的保存操作。
- ❑ 在 MATLAB 指令窗中，再次运行下列指令，便可得到如图 5.19 所示的图形。

```
K>> randn('seed',1),x=randn(1,100);barzzy1(x);
```

5.8　本章小结

MATLAB 除了本身提供大量可用的命令外，还提供了扩展开发的功能。可以根据需要编写相应功能的程序代码——M 文件。本章主要介绍了 M 文件的基础知识，包括数据类型、表达式和常见的程序结构。由于 MATLAB 的内核是由 C 语言编写的，所以如果熟悉 C 语言，会发现本章的内容十分熟悉。

本章涉及 MATLAB 脚本、函数（一般函数、内联函数、子函数、私用函数、方法函数）、程序调试和剖析并且配备了许多精心设计的算例，这些算例是完整的，可直接演练。读者通过这些算例，将真切感受到抽象概念的内涵、各指令间的协调，将从感知上领悟到面向对象编程的优越和至关要领。

5.9 习题

（1）命令文件与函数文件的主要区别是什么？

（2）找出 1～100 间 3 的倍数和尾数是 3 的数，按升序排列。

提示：排序函数为 sort(X)。

（3）编写脚本文件 Ex2.m 使用 while 循环计算从 1 开始的奇数的联乘积 S1，S1=1×3×5×…。要求 S1<1×106，显示 S1 和最后一个奇数的值。

（4）绘出函数 $y=\begin{cases} 2*x^2+1 & x>=1 \\ 0 & -1<x<1 \\ -x^3 & x<=-1 \end{cases}$ 的图像。

（5）建立一个命令 M 文件：求所有的“水仙花数”，所谓“水仙花数”是指一个三位数，其各位数字的立方和等于该数本身。例如，153 是一个水仙花数，因为 $153=1^3+5^3+3^3$。

（6）编写函数 M-文件 SQRT.m：用迭代法求 $x=\sqrt{a}$ 的值。求平方根的迭代公式为 $x_{n+1}=\frac{1}{2}(x_n+\frac{a}{x_n})$ 迭代的终止条件为前后两次求出的 x 的差的绝对值小于 10^{-5}。

第 6 章　Simulink 仿真

Simulink 是 MATLAB 的重要组成部分，它提供了集动态系统建模、仿真和综合分析于一体的图形用户环境。通过 Simulink 构造复杂仿真模型时，不需要书写大量的程序，只需要使用鼠标对已有模块进行简单操作，以及使用键盘设置模块的属性。它可以非常容易地实现可视化建模，并把理论研究和工程实践有机结合在一起，越来越受到人们的关注。本章将系统介绍 Simulink 的基本知识、常用模块集、子系统及其封装、模型仿真、模型调试和 S-函数等内容。

6.1　Simulink 介绍

Simulink 是 Math Works 公司为 MATLAB 提供的系统模型化的图形输入与仿真工具，它使仿真进入到了模型化的图形阶段。Simulink 主要有两个功能，即 Simu（仿真）和 Link（连接），它可以针对自动控制、信号处理及通信等系统进行建模、仿真和分析。

6.1.1　Simulink 概述

Simulink 是 MATLAB 的重要组成部分，它为用户提供了一个动态系统建模、仿真和综合分析的集成环境。在该环境中，无需大量书写程序，只需通过简单直观的鼠标操作，就可构造出复杂的系统。Simulink 同时支持线性和非线性、连续时间系统、离散时间系统、连续和混合系统建模且支持多进程，基于上述特点 Simulink 几乎可分析任何一种类型的真实动态系统。Simulink 具有适应面广、结构和流程清晰及仿真精细、贴近实际、效率高、灵活等优点，基于以上优点 Simulink 已被广泛应用于控制理论和数字信号处理的复杂仿真和设计。同时有大量的第三方软件和硬件可应用于或被要求应用于 Simulink。

Simulink 的特点如下。

（1）丰富的可扩充的预定义模块库。

（2）交互式的图形编辑器来组合和管理直观的模块图。

（3）以设计功能的层次性来分割模型，实现对复杂设计的管理。

（4）通过 Model Explorer 导航、创建、配置、搜索模型中的任意信号、参数、属性，生成模型代码。

（5）提供 API 用于与其他仿真程序的连接或与手写代码集成。

（6）使用 Embedded MATLAB 模块在 Simulink 和嵌入式系统执行中调用 MATLAB 算法。

（7）使用定步长或变步长运行仿真，根据仿真模式（Normal，Accelerator，Rapid Accelerator）来决定以解释性的方式运行或以编译 C 代码的形式来运行模型。

（8）图形化的调试器和剖析器来检查仿真结果，诊断设计的性能和异常行为。

（9）可访问 MATLAB 从而对结果进行分析与可视化，定制建模环境，定义信号参数和测试数据。

（10）模型分析和诊断工具来保证模型的一致性，确定模型中的错误。

本书用于演示的 Simulink 是包含在 MATLAB7.0 里的 Simulink6.0，启动 Simulink 有三种方式。

- 在 MATLAB 7.0 菜单栏中选择【File】/【New】/【Model】选项。
- 单击 MATLAB 7.0 工具栏上的 Simulink 按钮。
- 在 MATLAB 7.0 的命令窗口中直接键入“Simulink”命令。

运行后会弹出如图 6-1 所示的 Simulink 模块库浏览器窗口，使用第一种打开方式还会弹出如图 6-2 所示的新建模型窗口，使用第二种打开方式会弹出如图 6-1 所示窗口。单击图 6-1 工具栏中的图标（新建新模型）也会弹出如图 6-2 所示的窗口。使用第三种打开方式会弹出如图 6-3 所示窗口。

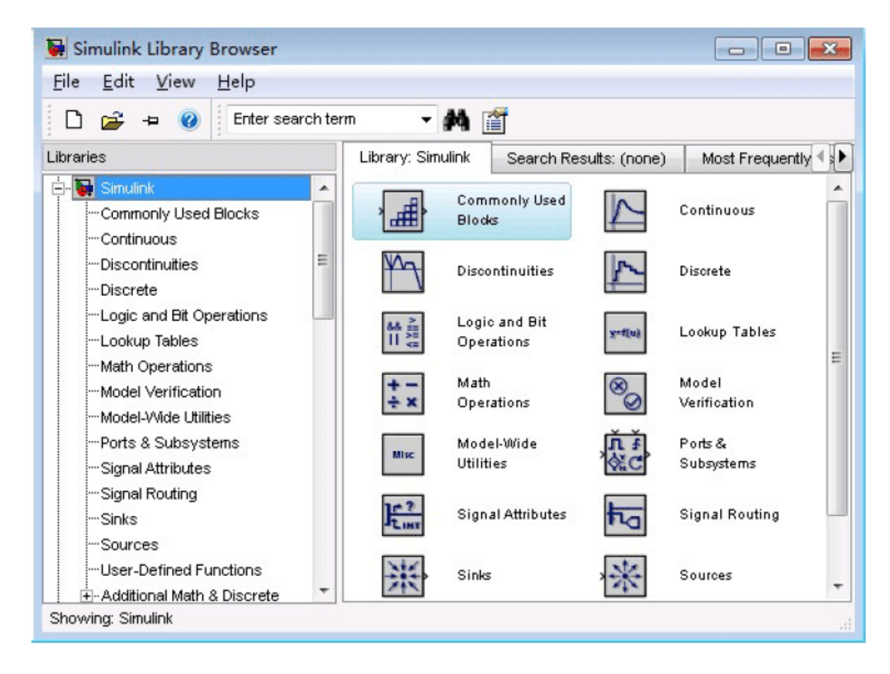


图 6-1　Simulink Library Browser 界面

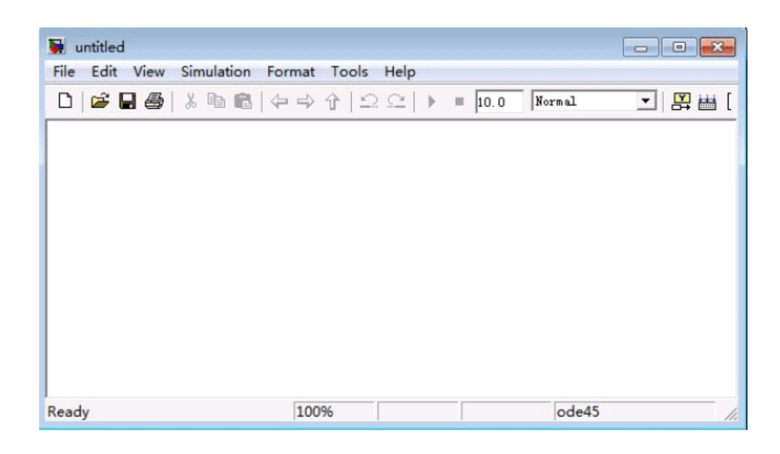


图 6-2　模型窗口

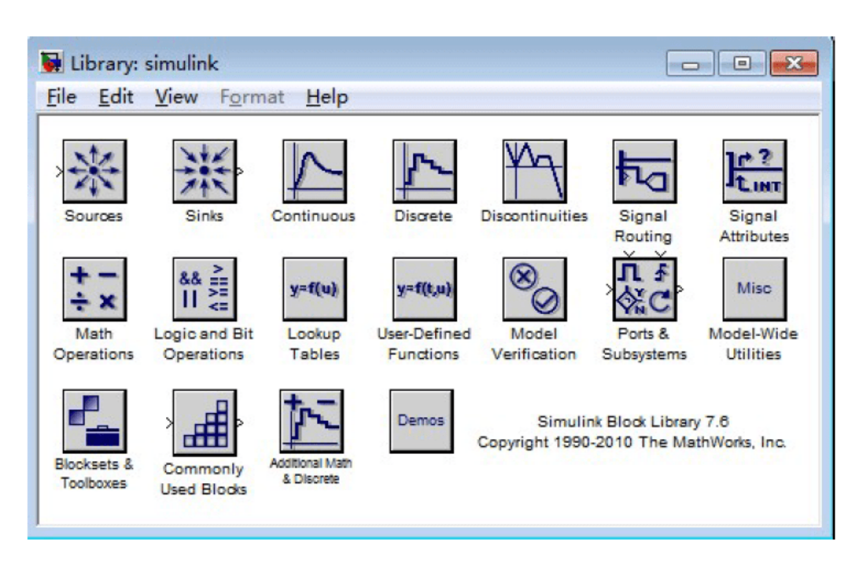


图 6-3　Library Simulink 界面

6.1.2　Simulink 工作环境

按照 6.1.1 节所介绍的方法启动 Simulink 后就可以看到如图 6-4 所示的 Simulink 模块库浏览器。由该浏览器可以看出 Simulink 模型库浏览器各部分的用途，包含公共模型库和专业模型库。

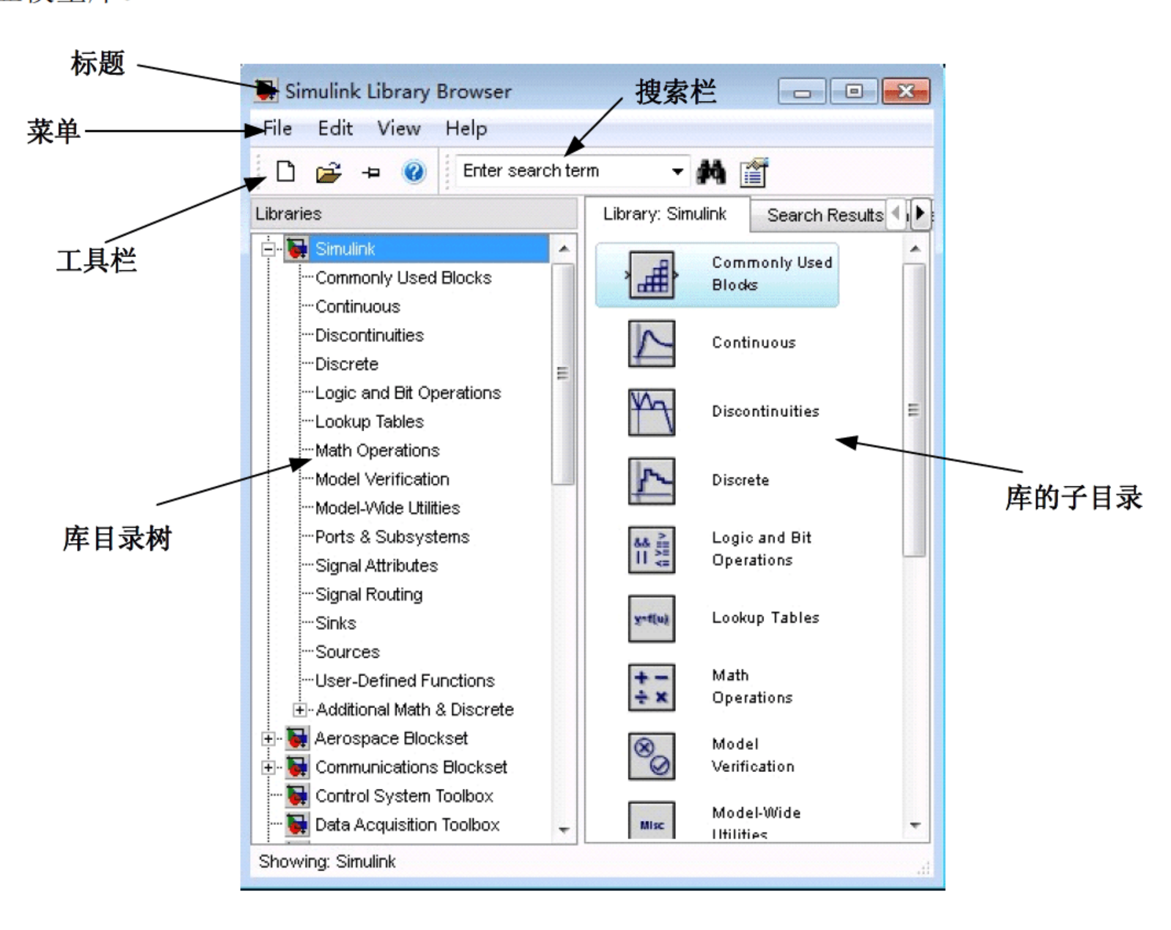


图 6-4

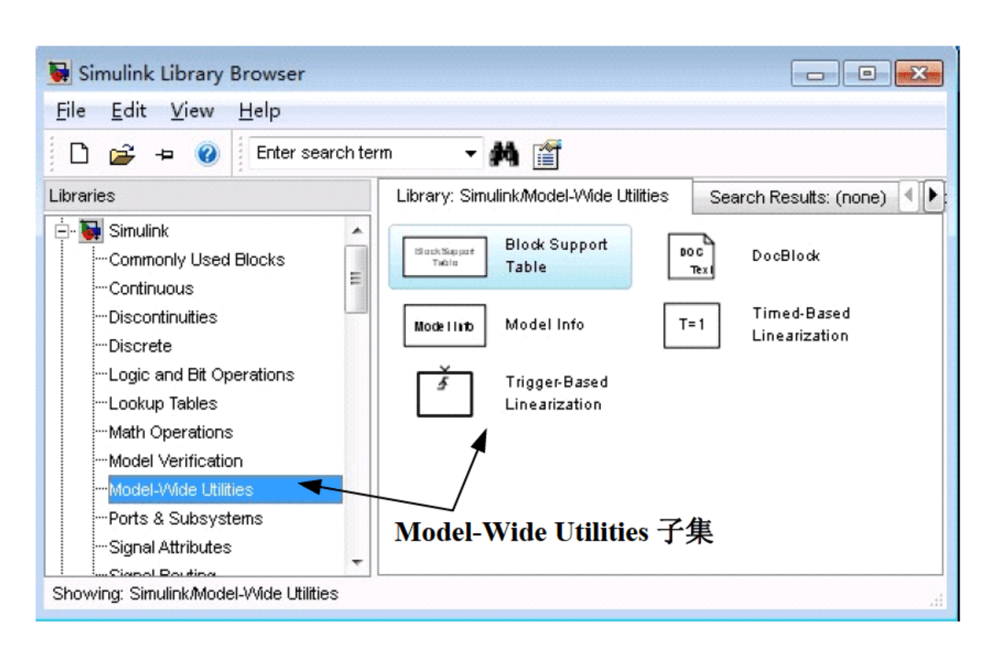


图 6-4　Simulink 模块库浏览器的结构

公共模块库中包含 16 个子模块库，它们分别为 Commonly Used Blocks（常用模型库）、Continuous（连续系统模型库）、Discontinuities（不连续循环模型库）、Discrete（离散系统模型库）、Logic and Bit Operations（逻辑及位操作库）、Lookup Tables（查表库）、Math Operations（数学运算库）、Model Verification（模型检验）、Model-Wide Utilities（针对模型的有用功能模块）、Ports & Subsystems（信号口与子系统）、Signal Attributes（信号特征库）、Signal Routing（信号路由库）、Sinks（输出方式库）、Sources（输入源）、User-Defined Functions（用户自定义函数库）、Additional Math & Discrete（数学离散模型库）。

6.1.3　Simulink 工作原理

Simulink 虽然提供了实现各种功能的模块，为用户屏蔽了许多繁琐的编程工作，但用户要想更加灵活高效地使用这个工具，就必须对其工作原理有一定的了解。Simulink 建模大致可分为两步：创建模型图标和控制 Simulink 对其进行仿真，但这些图像化模型和现实系统之间到底存在着什么样的映射关系，以及 Simulink 是如何对这些模型进行仿真的，下面就这两个问题予以说明。

1. 图形化的模型和现实系统间的映射关系

现实系统中都包含输入、状态和输出三个基本元素，以及三种元素间随时间变化的数学函数关系，在 Simulink 模型中每个图形化模块都可用图 6-5 表示，来代表现实系统中某个部分的输入、状态及输出随时间变化的函数关系，即系统的数学模型。系统的数学模型是由一系列的数学方程来描述，每一组数学方程都由一个模块来代表，Simulink 称这些方程为模块或模型的方法（一组 MATLAB 函数）。模块与模块间的连线代表系统中各元件输入/输出信号的连接关系，也代表了随时间变化的信号值。

通常，Simulink 模型的典型结构分为信源、系统和信宿三部分，其关系模型如图 6-6 所示。

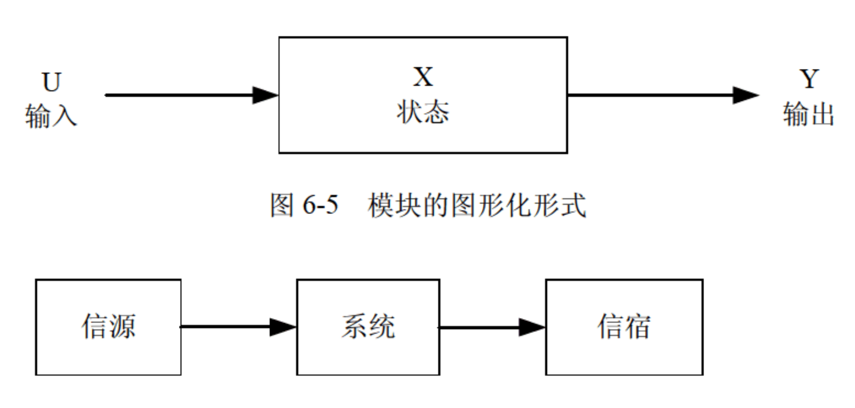


图 6-5　模块的图形化形式

图 6-6　Simulink 模型的典型结构

2. 利用映射关系进行仿真

在用户定义的时间段内根据模型提供的信息计算系统的状态和输出，并将计算结果予以显示和保存的过程，即是 Simulink 对模型进行仿真的过程。Simulink 的仿真过程包括如下几个阶段。

- ❑ 模型编译阶段。
- ❑ 连接阶段。
- ❑ 仿真环阶段。

1）模型编译阶段

Simulink 引擎调用模型编译器，将模型编译成可执行文件。编译器完成以下任务：计算模块参数的表达式以确定它们的值；确定信号属性（名字、数据类型等）；传递信号属性以确定未定义信号的属性；优化模块；展开模型的继承关系（如子系统）；确定模块运行的优先级；确定模块的采样时间。

2）连接阶段

Simulink 引擎创建按执行次序排列的运行列表，同时定位和初始化存储每个模块的运行信息。

3）仿真环阶段

Simulink 引擎从仿真的开始到结束，在每个采样点按运行列表计算各模块的状态和输出。仿真环阶段又可分为两个子阶段：一个是初始化阶段，此阶段只运行一次，用于初始化系统的状态和输出；第二个阶段为迭代阶段，该阶段在定义的时间段内按照采样点间的步长重复执行，用于在每个时间步计算模型的新的输入、状态和输出，并更新模型使之能反映系统最新的计算值。在仿真结束时，模型能反映系统最终的输入、状态和输出值。

6.2　Simulink 常用模块

Simulink 库浏览器窗口呈现一种树状结构，在其中列出了 Simulink 中的所有模块库，大体分为公共库和专业库，如 Simulink 库、Aerospace Blockset 库等，本节将介绍最常用的 Simulink 库中的一些子库，以便读者在阅读本书和学习中能够对所使用的模块有个初步的了解，下面主要介绍 Simulink 中常用子库中常用的模块的功能。

6.2.1 常用模块

从库目录树中选择【Simulink】/【Commonly Used Blocks】可得到如表 6-1 所示常用模块的模块名及其功能。

表 6-1 常用模块

模 块 名	功 能
Bus Creator	将输入信号合并成向量信号
Bus Selector	将输入向量分解成多个信号，输入只接受从 Mux 和 Bus Creator 输出的信号
Constant	输出常量信号
Data Type Conversion	数据类型转换
Demux	将输入向量转换成标量或更小的标量
Discrete-Time Integrator	离散积分器模块
Gain	增益模块
Ground	连接到其他块的没有连接的输入端口
In1	输入模块
Integrator	连续积分器模块
Logical Operator	逻辑运算模块
Mux	将输入的向量、标准量或矩阵信号合成
Out1	输出模块
Product	乘法器，执行标量、向量或矩阵的乘法
Relational Operator	关系运算，输出布尔类型数据
Saturation	定义输入信号的最大值和最小值
Scope	输出示波器

续表



模 块 名	功 能
Subsystem	创建子系统
Sum	加法器
Switch	选择器，根据第二个输入信号来选择输出第一个还是第三个信号
Terminator	终止输出，用于防止模型最后的输出端没有接任何模块时报错
Unit Delay	单位时间延迟
Vector Concatenate	将两个向量连接成一个向量

6.2.2 连续模块

从库目录树中选择【Simulink】/【Continuous】可得到如表 6-2 所示的连续模块的模块名及其功能。

表 6-2 连续模块

模 块 名	功 能
du/dt Derivative	连续微分器模块
$\frac{1}{s}$ Integrator	连续积分器模块
Integrator Limited	带限幅的积分器模块
Integrator, Second-Order	二阶积分器模块
Integrator, Second-Order Limit...	带限幅的二阶积分器模块
PID(s) PID Controller	PID 控制器
Ref PID(s) PID Controller (2DOF)	带设定值加权的 PID 控制器
x' = Ax+Bu y = Cx+Du State-Space	创建状态空间模型 dx/dt=Ax+Bu y=Cx+Du
$\frac{1}{s+1}$ Transfer Fcn	用矩阵形式描述的传输函数

模 块 名	功 能
Transport Delay	定义传输延迟，如果将延迟设置的比仿真步长大，可以得到更精确的结果
Variable Time Delay	可变时间延迟
Variable Transport Delay	定义传输延迟，第一个输入接收输入；第二个输入接收延迟时间
Zero-Pole	用矩阵描述系统零点，用向量描述系统极点和增益

6.2.3 非连续模块

从库目录树中选择【Simulink】/【Discontinuities】可得到如表 6-3 所示的非连续模块的模块名及其功能。

表 6-3 非连续模块

模 块 名	功 能
Backlash	间隙非线性
Coulomb & Viscous Friction	刻画在零点的不连续性，y=sign(x)*(Gain*abs(x)+Offest)
Dead Zone	产生死区，当输入在某一范围取值时输出为 0
Dead Zone Dynamic	产生死区，当输入在某一范围取值时输出为 0，与 Dead Zone 不同的是它的死区范围在仿真过程中是可变的，即动态死区非线性
Hit Crossing	检测输入是上升经过某一值还是下降经过这一值或是固定在某一值，用于过零检测
Quantizer	按相同的间隔离散输入，即量化非线性
Rate Limiter	限制输入的上升和下降速率在某一范围
Rate Limiter Dynamic	限制输入的上升和下降速率在某一范围，与 Rate Limiter 不同的是它的范围在仿真过程中是可变的
Relay	判断输入与某两域值的大小关系，当大于开启阈值时，输出为 on；当小于关闭阈值时，输出为 off；在两者之间时输出不变
Saturation	限制输入在最大和最小范围之内
Saturation Dynamic	限制输入在最大和最小范围之内，与 Saturation 不同的是它的范围在仿真过程中是可变的
Wrap To Zero	当输入大于某一值时输出 0，否则，输出等于输入

6.2.4 离散模块

从库目录树中选择【Simulink】/【Discrete】可得到如表 6-4 所示的离散模块的模块名及其功能。

表 6-4 离散模块

模 块 名	功 能
Difference	离散差分，输出当前值减去前一时刻的值
Discrete Derivative	离散偏微分
Discrete FIR Filter	离散 FIR 滤波器
Discrete Filter	离散滤波器
Discrete PID Controller	离散 PID 控制器
Discrete PID Controller (2DOF)	带设定值加权的离散 PID 控制器
Discrete State-Space	创建离散状态空间模型 x(n+1)=Ax(n)+Bu(n) y(n)=Cx(n)+Du(n)
Discrete Transfer Fcn	离散传输函数
Discrete Zero-Pole	离散零极点
Discrete-Time Integrator	离散积分器
First-Order Hold	一阶保持
Integer Delay	整数倍采样周期的延迟
Memory	存储单元，当前输出是前一时刻的输入
Tapped Delay	延迟
Transfer Fcn First Order	一阶传输函数，单位的直流增益
Transfer Fcn Lead or Lag	传递函数

续表

模　块　名	功　　能
Transfer Fcn Real Zero	离散零点传递函数
Unit Delay	一个采样周期的延迟
Zero-Order Hold	零阶保持

6.2.5　逻辑与位操作模块

从库目录树中选择【Simulink】/【Logic and Bit Operations】可得到如表 6-5 所示的逻辑与位操作模块的模块名及其功能。

表 6-5　逻辑与位操作模块

模　块　名	功　　能
Bit Clear	将向量信号中某一位置为 0
Bit Set	将向量信号中某一位置为 1
Bitwise Operator	对输入信号进行自定义的逻辑运算
Combinatorial Logic	组合逻辑，实现一个真值表
Compare To Constant	定义如何与常数进行比较
Compare To Zero	定义如何与零进行比较
Detect Change	检测输入的变化，如果输入的当前值与前一时刻的值不等，则输出 TRUE，否则为 FALSE
Detect Decrease	检测输入是否下降，是则输出 TRUE，否则输出 FALSE
Detect Fall Negative	若输入当前值是负数，前一时刻值为非负则输出 TRUE，否则为 FALSE
Detect Fall Nonpositive	若输入当前值是非正，前一时刻值为正数则输出 TRUE，否则为 FALSE
Detect Increase	检测输入是否上升，是则输出 TRUE，否则输出 FALSE
Detect Rise Nonnegative	若输入当前值是非负，前一时刻为负数则输出 TRUE 否则为 FALSE
Detect Rise Positive	若输入当前值是正数，前一时刻为非正则输出 TRUE 否则为 FALSE

续表



模 块 名	功 能
Extract Bits	从输入中提取某几位输出
Interval Test	检测输入是否在某两个值之间，是则输出 TRUE 否则输出 FALSE
Interval Test Dynamic	测试动态时间间隔
Logical Operator	逻辑运算
Relational Operator	关系运算
Shift Arithmetic	算数运算

6.2.6 查找表模块

从库目录树中选择【Simulink】/【Lookup Tables】可得到如表 6-6 所示的查找表模块的模块名及其功能。

表 6-6 查找表模块

模 块 名	功 能
Cosine	余弦函数查询表
Direct Lookup Table (n-D)	n 个输入信号的查询表
Interpolation Using Prelookup	n 个输入信号的预插值
Lookup Table	输入信号的查询表
Lookup Table (2-D)	二维输入信号的查询表
Lookup Table (n-D)	n 维输入信号的查询表
Lookup Table Dynamic	动态查询表
Prelookup	预查询
Sine	正弦函数查询表

6.2.7 数学模块

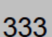
从库目录树中选择【Simulink】/【Math Operations】可得到如表 6-7 所示的数学模块的模块名及其功能。

表 6-7 数学模块

模 块 名	功 能
Abs	求绝对值
Add	加法运算
Algebraic Constraint	将输入约束为零，主要用于代数等式的建模
Assignment	选择输出输入的某些值
Bias	将输入加一个偏移，y=u+Bias
Complex to Magnitude-Angle	将输入的复数转换成幅度和幅角
Complex to Real-Imag	将输入的复数转换成实部和虚部
Divide	实现除法或乘法
Dot Product	点乘运算
Find Nonzero Elements	求线性指数或非零输入值的下标
Gain	增益，实现点乘或普通乘法，即比例运算
Magnitude-Angle to Complex	将输入的幅度和幅角合成复数输出
Math Function	实现数学函数运算
Matrix Concatenate	实现矩阵的串联
MinMax	将输入的最小或最大值输出，即最值运算
MinMax Running Resettable	最小和最大值运算，当有重置信号 R 输入时，输出被重置为初始值
Permute Dimensions	重新安排元素位置

续表

模 块 名	功 能
Polynomial	多项式求值，多项式的系数以数组的形式定义
Product	乘运算
Product of Elements	将所有输入实现连乘
Real-Imag to Complex	将输入的两个数当成一个复数的实部和虚部合成一个复数
Reciprocal Sqrt	平方根函数的倒数
Reshape	改变输入信号的维数
Rounding Function	将输入的整数部分输出
Sign	判断输入的符号。若为正则输出 1，为负输出-1，为 0 输出 0
Signed Sqrt	符号平方根函数
Sine Wave Function	产生一个正弦函数
Slider Gain	可变增益
Sqrt	平方根函数
Squeeze	去除数组中长度为 1 的维
Subtract	减法运算
Sum	加法运算
Sum of Elements	元素的和运算
Trigonometric Function	三角函数，包括正弦、余弦、正切和余切等
Unary Minus	一元减法
Vector Concatenate	数组数据的合成
Weighted Sample Time Math	根据采样时间实现输入的加法、减法、乘法和除法，只对离散信号适用



6.2.8　信号接收器模块

从库目录树中选择【Simulink】/【Sinks】可得到如表 6-8 所示的信号接收器模块的模块名及其功能。

表 6-8　信号接收器模块

模　块　名	功　　能
Display	显示输入数值的模块
Floating Scope	浮置示波器，由用户来设置所要显示的数据
Out1	输出模块
Scope	示波器
Stop Simulation	当输入不为零时，停止仿真
Terminator	连接到未连接的输入端口
To File	将输入和时间写入 MAT 文件
To Workspace	将输入和时间写入 MATLAB 工作空间中的数组或结构中
XY Graph	将输入分别当成 X，Y 轴数据绘制成二维图形

6.2.9　信号源模块

从库目录树中选择【Simulink】/【Sources】可得到如表 6-9 所示的信号源模块的模块名及其功能。

表 6-9　信号源模块

模　块　名	功　　能
Band-Limited White Noise	有限带宽的白噪声
Chirp Signal	产生 Chirp 信号
Clock	输出当前仿真时间
Constant	输出常数

续表

模块名	功能
Counter Free-Running	自动计数器，发生溢出后又从 0 开始
Counter Limited	有限计数器，当计数到某一值后又从 0 开始
Digital Clock	以数字形式显示当前的仿真时间
Enumerated Constant	输出枚举型常数
From File	从 MAT 文件中读取数据
From Workspace	从 MATLAB 工作空间读取数据
Ground	接地
In1	输入信号
Pulse Generator	产生脉冲信号
Ramp	产生按某一斜率的数据，即斜坡信号输入
Random Number	产生正态分布的随机数
Repeating Sequence	重复输出某一数据序列
Repeating Sequence Interpol...	重复序列内插值
Repeating Sequence Stair	重复阶梯序列
Signal Builder	信号创建器
Signal Generator	信号发生器
Sine Wave	产生正弦波信号
Step	产生阶跃信号
Uniform Random Number	按某一分布在某一范围内生成随机数



6.2.10 用户自定义函数模块

从库目录树中选择【Simulink】/【User-Ds】可得到如表 6-10 所示的用户自定义函数模块的模块名及其功能。

表 6-10 用户自定义函数模块

模 块 名	功 能
Embedded MATLAB Function	内置 MATLAB 函数模块，在模型窗口中双击该模块图标就会弹出 M 文件编辑器
Fcn	简单的 MATLAB 函数表达式模块
Level-2 MATLAB S-Function	Level-2 型 M 文件 S-函数
MATLAB Fcn	用于指定自编函数
S-Function	用于指定 S-函数
S-Function Builder	具有 GUI 界面的 S-函数创建器
S-Function Examples	S-函数演示模块，在模型窗口中双击该模块图标可以看到多个 S 函数示例

除了以上举例的模块外，在 Simulink 模块库浏览器下拉菜单中，还有许多其他模块，用户可根据需要查看各个模块的功能。

下面将结合具体实例来介绍一些模块的使用方法，以便读者能对 Simulink 有更进一步的了解。添加一个模块，只需要在模块浏览器中找到该模块，选中并拖放到模型窗口即可；删除一个模块，只需要在模型窗口中选中并删除即可；模块之间的连接比较简单，只需要选中一个模块的输出端，然后用鼠标拖动到另一个模块的输入端即可，或首先选中一个模块的输出端，然后用鼠标拖动到已经存在的连线上即可；设置模块参数只需双击指定模块，然后设置相应参数项即可。

【例 6-1】 使用常用的信号接收器和信号源模块。

（1）首先建立如图 6-7 所示的模型并保存，其中各模块的类型和名称为其下方的提示文字，其次按如下方式设置模块的参数。

- **Clock 模块** 选中用于显示时间的参数 Display Time 并调整模块大小。
- **Scope 模块** 如图 6-8 所示，勾掉用于设定保存数据个数的参数“Limit data points last”，否则，只保存最后 5 000 个数据（其中 5 000 这个参数是可以进行设置的）。
- **Step 模块** 设置终值参数 Final value 为 2。
- **To Workspace 模块** 变量名称参数“Variable Name”默认值为“simout”。
- **Sine Wave 模块** 如图 6-9 所示设置频率参数“Frequency”为“pi/4”和偏置角参数“Phase”为“pi/2”。

- ❑ **Compare To Constant 模块** 如图 6-10 所示设置操作参数“Constant value”为“0.2”和输出类型参数“Output data type mode”为“boolean”，该模块实现的功能是当输入大于 0.2 时输出为 0，否则为 1。
- ❑ **Stop Simulink 模块** 遇到非零值则停止仿真。
- ❑ **From Workspace 模块** 变量名称参数“Variable Name”默认值为“simin”。
- ❑ **Terminator 模块** 避免其他模块的输出端口不连接而导致的警告信息。

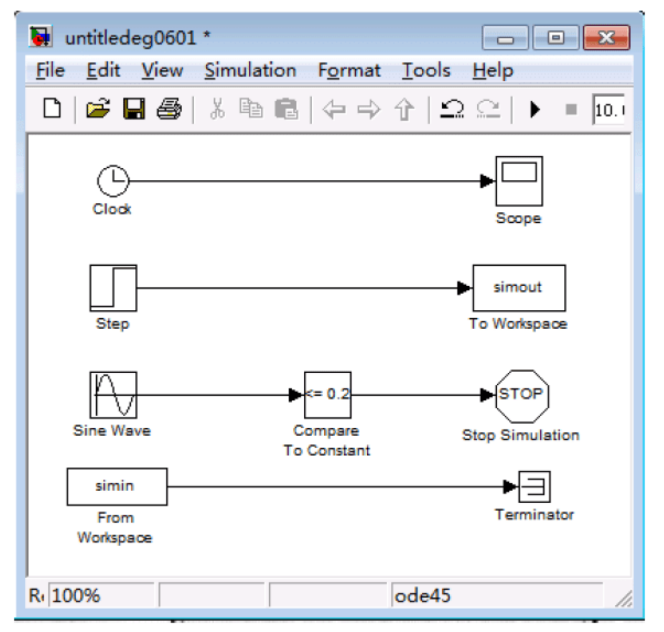


图 6-7 Simulink 模块

图 6-8 Scope 模块设置

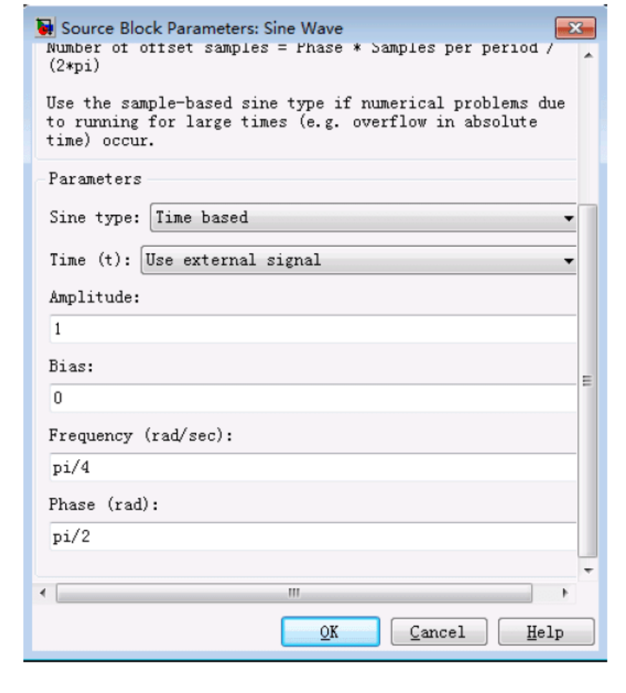


图 6-9 Sine Wave 模块设置

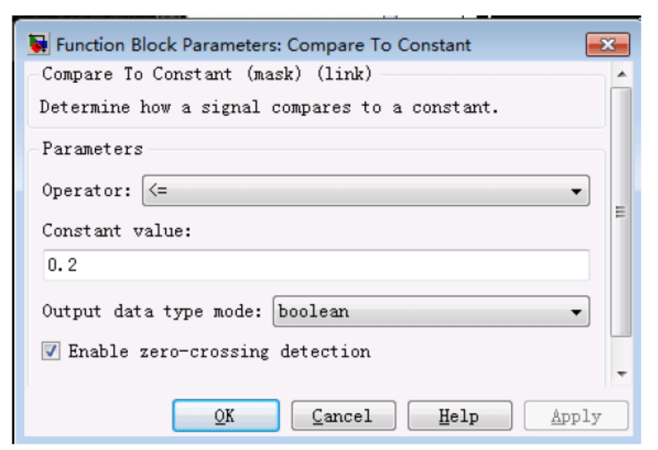


图 6-10 Compare To Constant 模块设置

（2）再次在命令窗口中输入如下语句，以便为变量 simin 赋值。

```
clear
clc
simin(:,1)=[0:10]';
simin(:,2)=rand(11,1);
```

（3）最后运行该模型，双击打开 Scope 模块并通过鼠标左键选择“Autoscale”选项，可以得到如图 6-11 所示的结果。

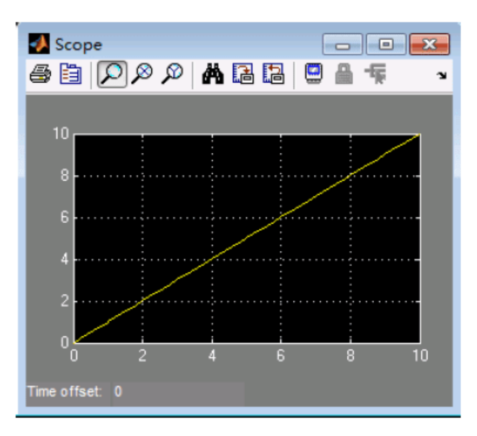


图 6-11 Scope 模块运行结果

（4）同时在命令窗口输入语句 simout，命令窗口中的输出结果如下。

```
simout =
          time: []
       signals:[1x1 struct]
     blockName:'untitledeg0601/To Workspace'
```

显然，simout 为结构体类型，继续在命令窗口输入如下语句：Y=simout.signals，命令窗口中的输出结果如下。

```
Y=
       values:  [53x1  double]
   dimensions:  1
        label:  ''
```

显然，Y 仍为结构体类型，接着在命令窗口输入语句：YY=Y.values'，命令窗口中的输出结果如下。

```
YY=
     0     0     0     0     0     0     2     2     2     2     2     2
```

由此读出“To Workspace”模块保存的数据，该模块还可以将数据保存为带有时间信息的结构体类型和数组类型，其中对于数组类型读取数据可能更方便一些。

6.3 Simulink 其他模块

Simulink 可以针对控制、信号处理及通信等系统进行建模、仿真和分析，因此除了前

面介绍的常用模块集之外，还提供了其他模块集和工具箱。下面通过具体例子对其中几个模块集或工具箱中的模块进行介绍。

【例 6-2】 使用 Simulink Extras 模块集中带有初始参数的传递函数模块。

（1）建立如图 6-12 所示的模型并保存。

需要说明的是：

- ❑ Transfer Fcn（with initial outputs）模块的初始输出和参数设置如图 6-13 所示。
- ❑ Transfer Fcn（with initial states）模块的初始状态和参数设置如图 6-14 所示。
- ❑ Spectrum Analyzer 模块为系统频谱分析器，它的第一个输入表示系统输入信号，第二个输入表示系统输出信号。

（2）运行该模型，然后双击 Spectrum Analyzer 模块可得到如图 6-15 所示的结果，双击 Scope 模块可得到如图 6-16 所示的结果。

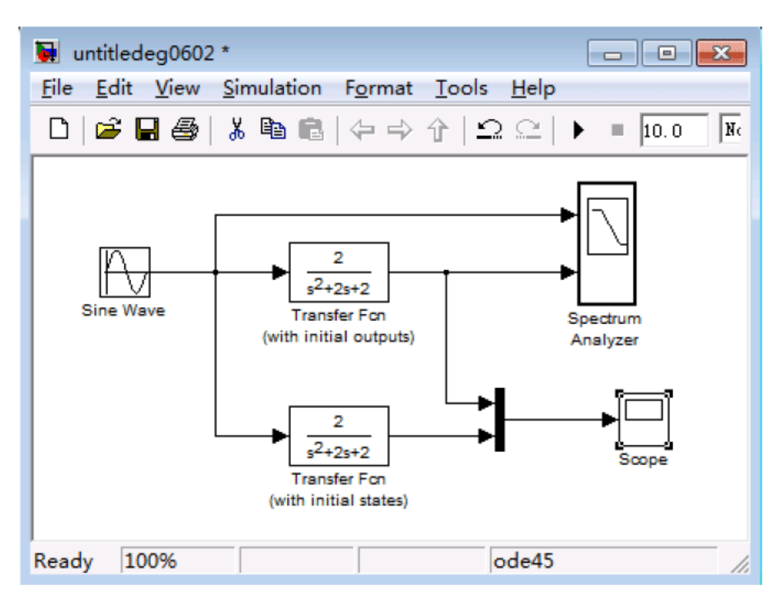


图 6-12 Simulink 模型

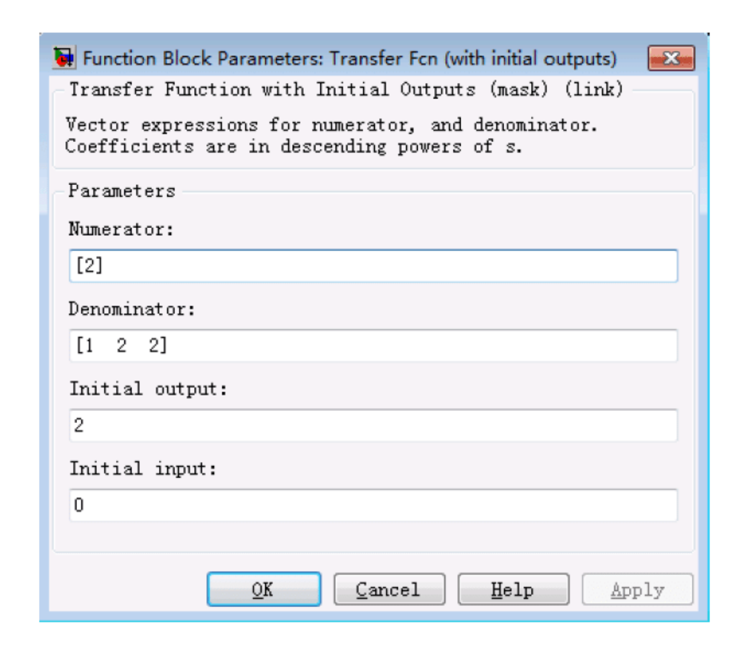


图 6-13 Transfer Fcn（with initial outputs）模块设置



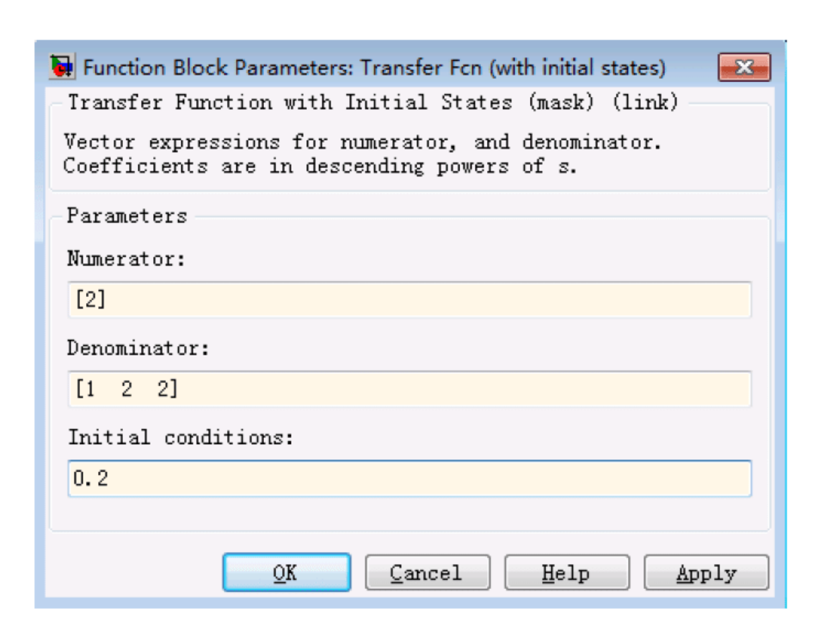


图 6-14 Transfer Fcn（with initial states）模块设置

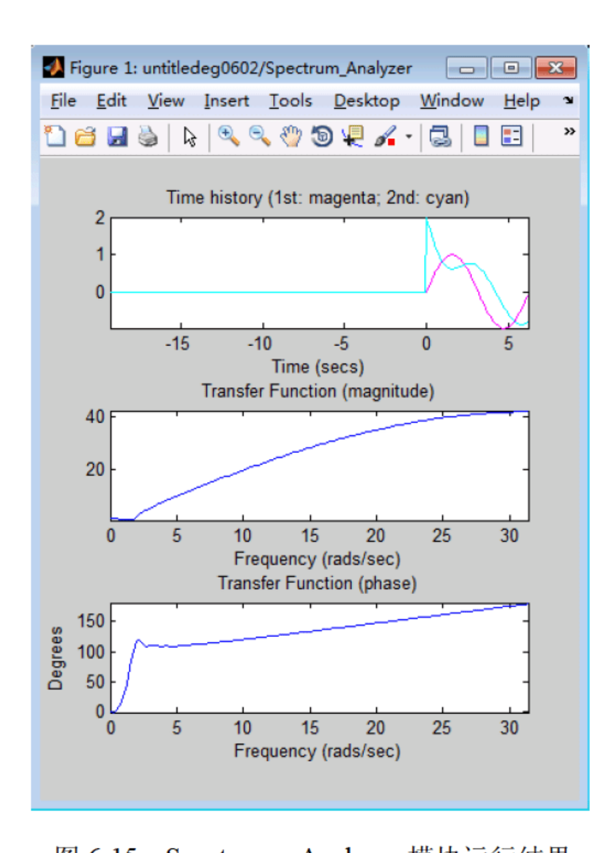


图 6-15 Spectrum Analyzer 模块运行结果

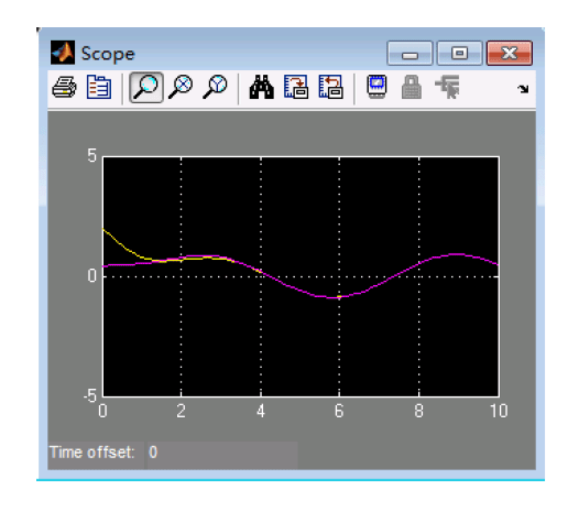


图 6-16 Scope 模块运行结果

6.4 Simulink 模型创建

前面几节的讲解使我们对 Simulink 已经有了初步的认识，下面来学习如何创建 Simulink 模型，如图 6-17 所示为建立 Simulink 模型的流程图。

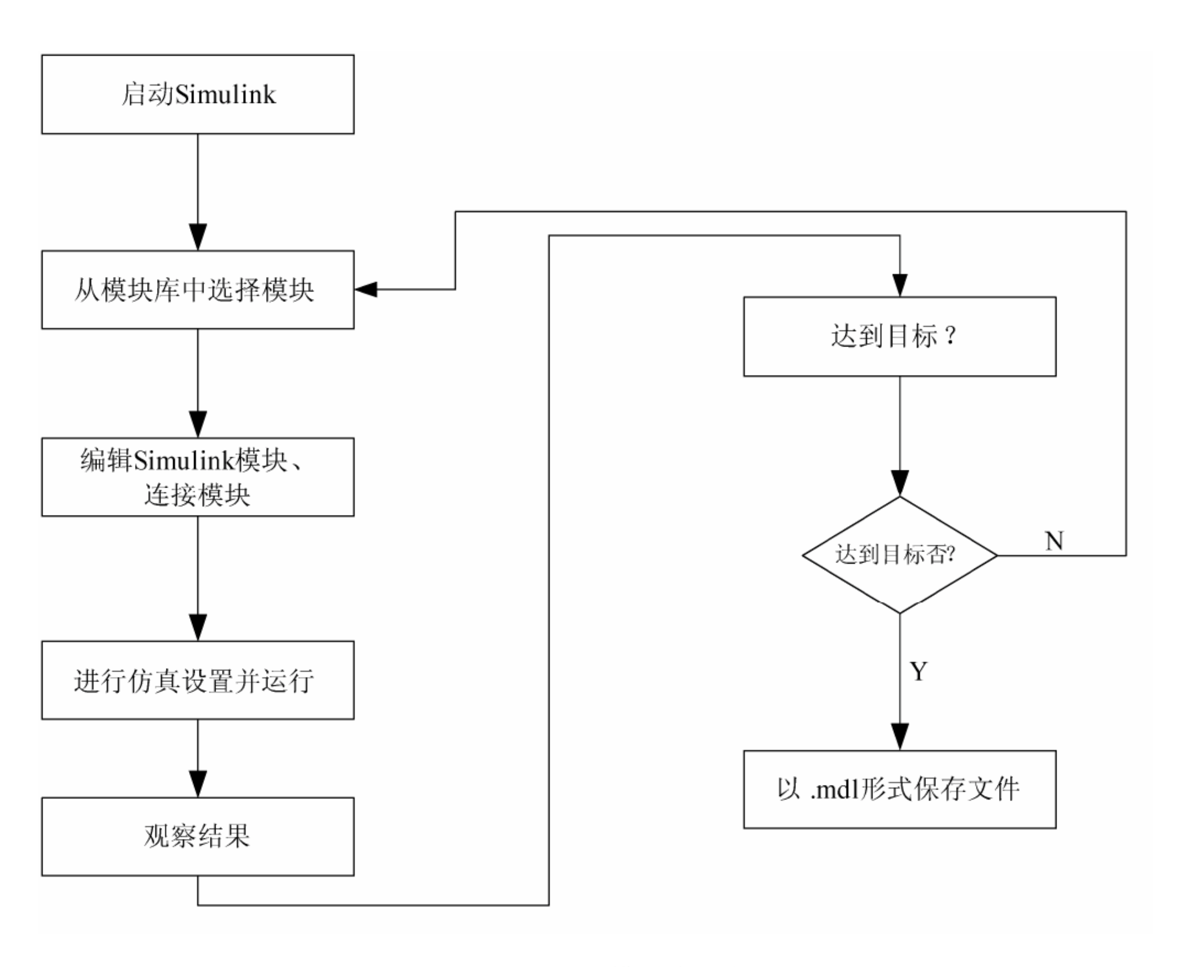


图 6-17　Simulink 模型建立流程图

6.4.1　模块操作

Simulink 模块操作包括选择一个或多个模块，复制、删除和移动模块，模块外形的调整，模块名的操作，定义模块中的参数和属性，模块间的连接等。

1. 模块的选择

选择模块有两种情况，即选择一个模块和选择多个模块。

1）选择一个模块

选择一个模块只需要使用鼠标左键单击指定模块，当用户选中一个模块时，以前选中的模块就被放弃。

2）选择多个模块

选择多个模块可以有两种方法：一个是逐个选择法；另一个是使用方框选择相邻的几个模块。

- **逐个选择法**　按住 Shift 键，使用鼠标左键单击需要选中的模块。
- **方框选择法**　使用鼠标单击和拖动以画出方框，选择方框内的所有模块。

2. 复制、删除和移动模块

1）复制模块

- **不同窗口复制模块**　选中模块后，直接将模块从一个窗口拖动到另一个窗口即可。

- **同一个窗口内复制模块** 选中模块后，按下快捷键 Ctrl+C 实现复制，再按下快捷键 Ctrl+V 实现粘贴。

> 📖 实现粘贴，还可以通过菜单 Edit 中的 copy 和 Paste 来复制模块。

2）删除模块

删除模块可以采用以下两种方法。

- 选中后，按 Delete 键删除模块。
- 选中模块后，通过 Edit 菜单中的 Cut 和 Delete 来删除模块。

3）移动模块

按住鼠标左键直接将模块拖动到指定位置。

3. 模块外形的调整

模块外形的调整包括三种形式，即改变模块的大小、调整模块的方向和给模块添加阴影。

1）改变模块的大小

选中模块后，将鼠标移动到模块边框的一角，当鼠标变成两端有箭头的线段时，按下鼠标左键拖动模块图标来改变模块大小。

2）调整模块的方向

选中模块后，通过菜单【Format】/【Rotate Block】使模块水平方向顺时针旋转 90°，通过菜单【Format】/【Flip Block】使模块相对于垂直方向反正 180°。

如图 6-18 所示中间的模块是最初的模块，左边的 4 个模块从上到下依次通过菜单【Format】/【Rotate Block】使模块旋转一次、两次、三次和四次；右边的两个模块从上到下依次通过菜单【Format】/【Flip Block】使模块旋转一次和两次；下方的模块是先通过菜单【Format】/【Flip Block】使模块旋转一次，再通过菜单【Format】/【Rotate Block】使模块旋转一次。

3）给模块添加阴影

选中模块后，通过【Format】/【Show Drop Shadow】给模块添加阴影。

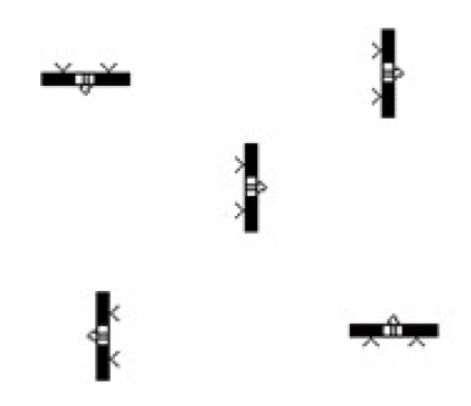


图 6-18 模块的旋转

4. 模块名的操作

模块名的操作包括修改模块名、显示模块名和改变模块名的位置。

1）修改模块名

通过鼠标左键双击模块名，修改模块名。

2）显示模块名

选中模块后，通过菜单【Format】/【ShowName】来显示模块名；通过菜单【Format】/【Hide Name】来隐藏模块名。

3）改变模块名的位置

选中模块后，通过菜单【Format】/【Flip Name】来改变模块名的显示位置。

5. 定义模块中的参数

定义模块中的参数有以下三种方法。

- ❑ 用户通过双击需要设置参数的模块，得到如图 6-19 所示的模块参数设置对话框，定义模块中的参数。
- ❑ 用鼠标右键单击模块并在弹出的菜单中选择“Transfer Fcn Parameters...”，定义模块中的参数。
- ❑ 选择要设置的模块后，选择菜单【Edit】/【Transfer Fcn Parameters...】，定义模块中的参数。

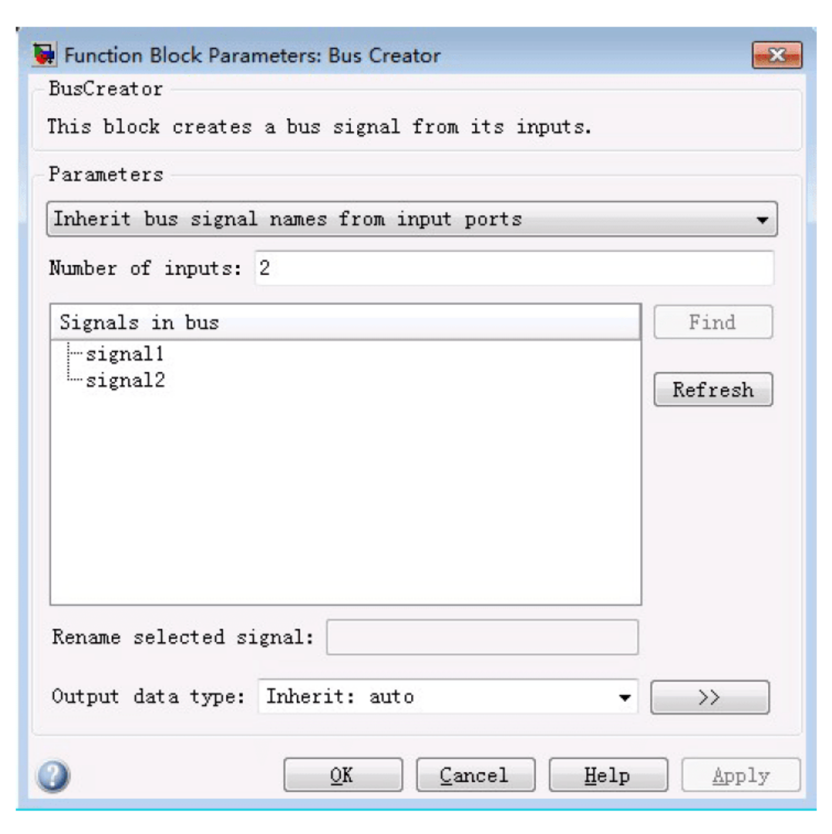


图 6-19　模块参数设置对话框

6. 定义模块的属性

Simulink 中的每个模块都有一个内容同如图 6-20 所示的属性设置对话框，可以通过两种方式打开此属性设置对话框。

- ❑ 用鼠标右键单击模块并在弹出的菜单中选择“Block Properties...”命令。
- ❑ 选中要设置的模块后，选择菜单【Edit】/【Block Properties】命令。

图 6-20　属性设置对话框

7. 模块的连接

模块之间的连接一般是通过直线完成的，下面分别在表 6-11～表 6-13 中介绍 Microsoft Windows 环境下对直线操作、直线信息和注释文字的处理。

表 6-11　直线操作

任　务	Microsoft Windows 环境下的操作
选择多条直线	与选择多个模块的方法相同
选择一条直线	单击要选择的直线，当用户选择一条直线时，之前选择的直线被放弃
连线的分支	按下 Ctrl 键，然后拖动直线
移动直线	按下鼠标左键直接拖动直线
移动直线定点	将鼠标指向连线的箭头处，当出现一个小圆圈圈住箭头时按下鼠标左键并移动连线
直线调整为斜线段	按下 Shift 键，将鼠标指向需要移动的直线上的一点并按下鼠标左键直接拖动直线
直线调整为直线段	按住鼠标左键不放直接拖动直线

表 6-12　直线信息处理

任　务	Microsoft Windows 环境下的操作
建立信号标签	在直线上双击，然后输入标签
复制信号标签	按下 Ctrl 键，然后按下鼠标左键选中标签并拖动
移动信号标签	按下鼠标左键选中标签并拖动
编辑信号标签	在标签框内双击，然后进行编辑

续表

任　　务	Microsoft Windows 环境下的操作
删除信号标签	按下 Shift 键，然后用鼠标选中标签，再按 Delete 键
用粗线表示向量	选择【Foamat】/【Port/Signal Display】/【Wide Nonscalar Lines】菜单命令
显示数据类型	选择【Foamat】/【Port/Signal Display】/【Port Data Types】菜单命令

表 6-13　注释文字处理

任　　务	Microsoft Windows 环境下的操作
建立注释	在模型图标中双击，然后输入文字
复制注释	按下 Ctrl 键，然后按下鼠标左键选中注释文字并拖动
移动注释	按下鼠标左键选中注释文字并拖动
编辑注释	单击注释文字，然后进行编辑
删除注释	按下 Shift 键，然后用鼠标选中注释文字，再按 Delete 键

6.4.2　基本步骤

通过对前面内容的讲解，读者可以了解 Simulink 的一些基础知识。下面总结使用 Simulink 进行系统建模和仿真的步骤。

（1）画出系统框图，将要仿真的系统根据功能划分成子系统，然后选用模块来搭建每个子系统。

（2）启动 Simulink 模块库浏览器，新建一个空白模型。

（3）在模块库中找到所需模块并拖曳到空白模型窗口中，按系统框图的布局摆放好各模块并连接各模块。

（4）如果系统比较复杂，模块的数目太多，用户可以将同一功能的模块封装成一个子系统。

（5）设置各模块的参数及与仿真有关的各种参数。

（6）将模型保存为后缀名为.mdl 的模型文件。

（7）运行仿真，观察结果。如果仿真出错，请按弹出的错误提示框来查看出错误原因，进行修改；如果仿真结果与预想的结果不符，首先检查模块的连接是否有误，选择的模块是否合适，然后检查模块参数和仿真参数的设置是否合理。

（8）调试模型。若在第（7）步中没有出现任何错误提示，但是仿真结果与预想的结果不符，就需要进行调试，来查看系统在每个采样点的运行情况，以便找到导致仿真结果与预想情况或实际情况不符的地方。修改后再仿真，直到结果符合要求为止。

6.4.3　Simulink 简单建模仿真示例

Simulink 最大的功能在于其建模仿真功能，在本节中，将通过一个具体实例展示 Simulink 到底能够做什么，通过本实例，读者就可以直接建立自己简单的 Simulink 模型。在本例中的模型有如下要求。

❑ 信号源为脉冲信号和正弦信号。

❑ 将脉冲信号放大两倍，正弦信号不变。

❑ 将放大后的脉冲信号和正弦信号在同一示波器中输出。

在 Simulink 中，建模步骤如下。

（1）打开 MATLAB 界面，单击工具栏中的按钮打开 Simulink 库浏览器，如图 6-1 所示。

（2）单击 Simulink 库浏览器工具栏中的按钮，打开一个默认名称为 untitled 的空白模型窗口，如图 6-21 所示。

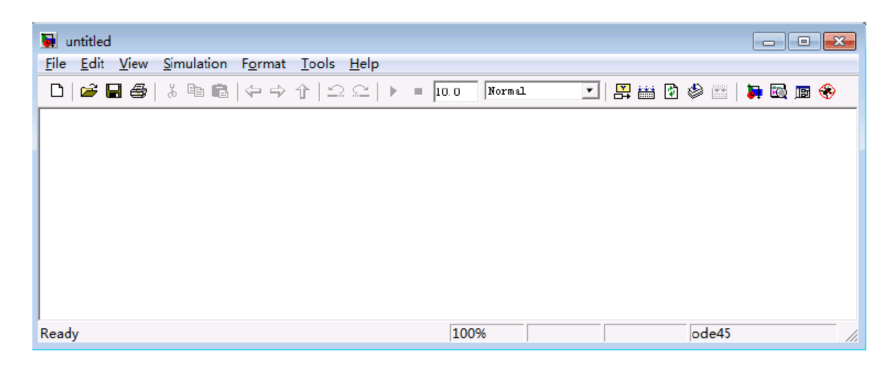


图 6-21　Simulink 模型窗口

（3）在模型窗口中添加信号源，在 Simulink 库浏览器中单击 Simulink 库中的子库 Sources，在右侧展开列表框中分别选择 Sine Wave（正弦波）模块和 Pulse Generator（脉冲发生器）模块，按下鼠标左键的同时拖动模块到新建的空白模型窗口中，释放鼠标左键，即可分别将正弦波模块和脉冲发生器模块从库浏览器中复制到模型窗口中，如图 6-22 所示。

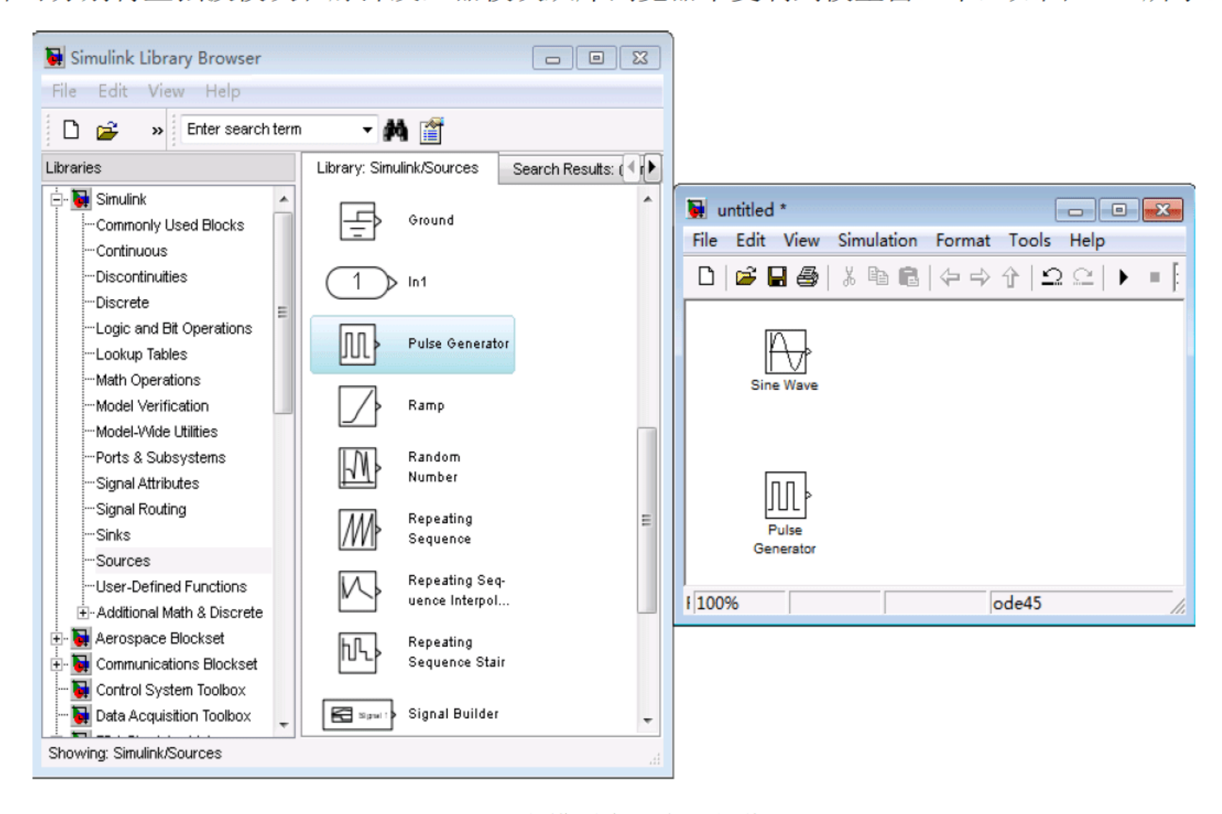


图 6-22　向模型窗口中添加信号源

（4）按照步骤（3）的方法，在模型窗口中添加其他模块。将 Simulink 库中的子库 Math Operations 中的 Gain 模块、Signal Routing 子库中的 Mux 模块和 Sink 子库中的 Scope 模块一次拖动到新建的模型窗口中，如图 6-23 所示。Gain 模块用来放大脉冲信号；Mux 模块用来将放大后的脉冲信号和正弦信号两路信号复合为一路信号输出；Scope 模块用来显示输出信号。

（5）连接模块。将鼠标指向模块的输出端，当光标变为十字形符号时，按下鼠标左键，移动鼠标至要连接模块的输入端，当光标由十字形符号变为双十字形符号时松开鼠标，即可完成两个模块间的连接，最终完成所有模块间的连接，如图 6-24 所示。

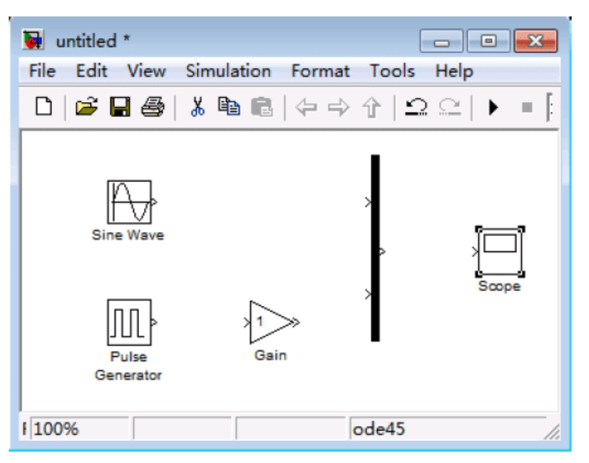


图 6-23　向模型窗口中添加其他模块

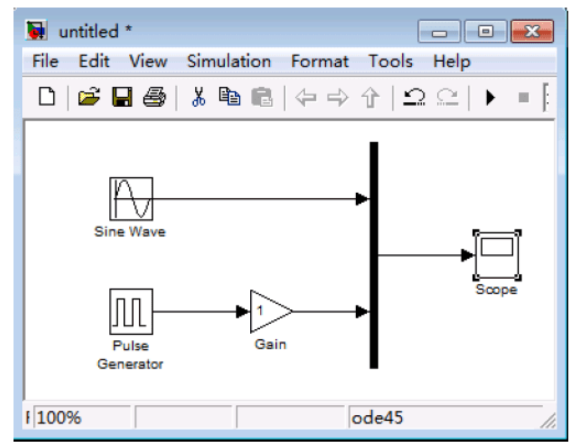


图 6-24　连接模块

（6）单击模型窗口工具栏中的按钮，保存模型，弹出如图 6-25 所示的保存窗口，将模型名称改为 Sample，单击“保存”按钮保存。

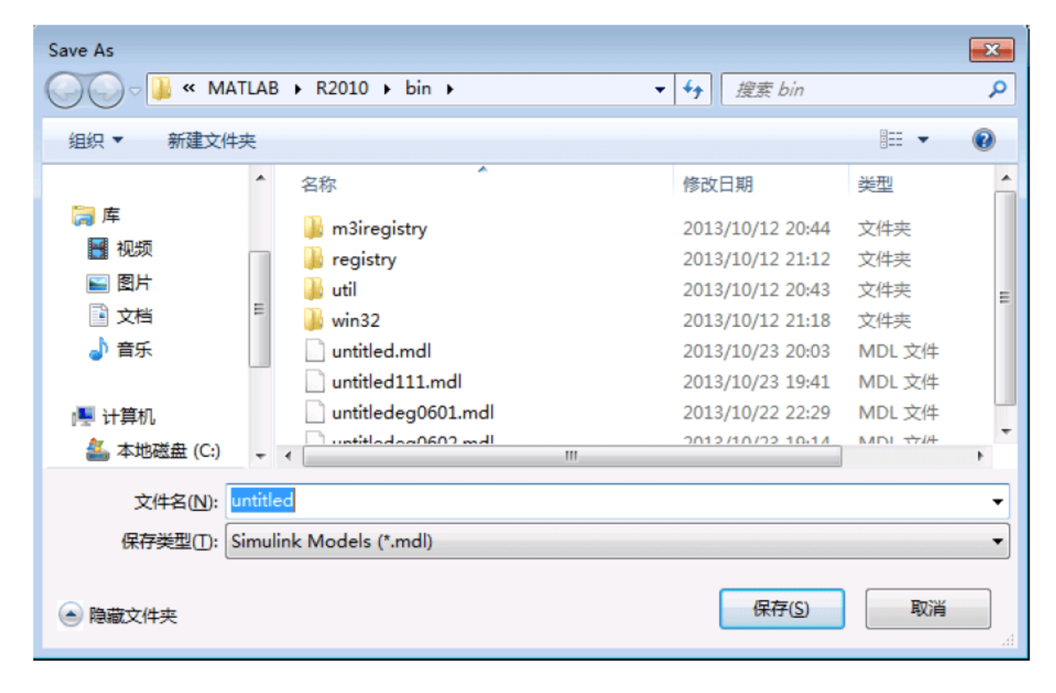


图 6-25　保存模型

（7）设置模块属性。双击 Gain 模块，将弹出 Gain 模块参数设置对话框，设置参数 Gain

为 2，如图 6-26 所示，设置完毕之后单击 OK 按钮关闭对话框。其他模块采用默认设置，在本例中不做修改。

（8）显示仿真结果。单击模型窗口中的“仿真”按钮▶，开始仿真。在示波器上，可以观察到原信号和放大后的输出信号。单击示波器工具栏中的“自动刻度”按钮🔭，使得波形充满整个坐标框，得到如图 6-27 所示的图形。

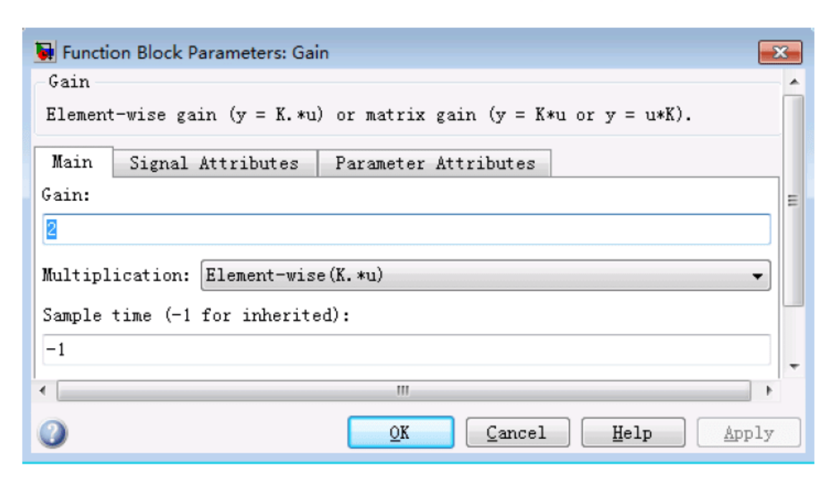


图 6-26　模块参数设置

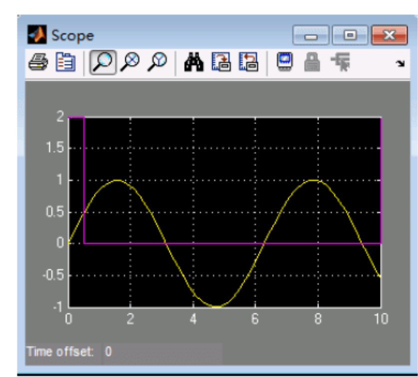


图 6-27　示波器输出结果显示

6.5　子系统及其封装

随着系统规模的不断扩大，复杂性不断增加，模型的结构也变得越来越复杂。在这种情况下，将功能相关的模块组合在一起形成几个小系统，将使整个模型变得非常简洁，使用起来非常方便。下面分别介绍子系统的创建和封装方法。

6.5.1　子系统的创建

1. 子系统的作用

通过子系统可以把复杂的模型分割成若干个简单的模型，具有以下优点。

（1）减少模型窗口中模块的个数，使得模型窗口整洁。

（2）把一些功能相关的模块集成在一起，可以复用。

（3）通过子系统可以实现模型图表的层次化。

2. 子系统的创建方法

Simulink 有如下两种创建子系统的方法。

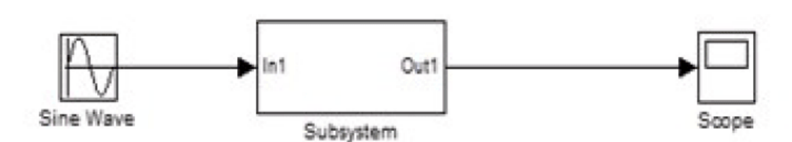
（1）通过子系统模块来创建子系统：先向模型中添加 Subsystem 模块，然后打开该模块并向其中添加模块；

（2）组合已存在的模块集。

3. 子系统创建示例

【例 6-3】 通过 Subsystem 模块创建子系统，具体步骤如下。

（1）从 Ports&Subsystems 中复制 Subsystem 模块到模型中，如图 6-28 所示。
（2）双击 Subsystem 模块图标即可打开 Subsystem 模块编辑窗口。
（3）在新的空白窗口创建子系统，然后保存。
（4）运行仿真并保存。



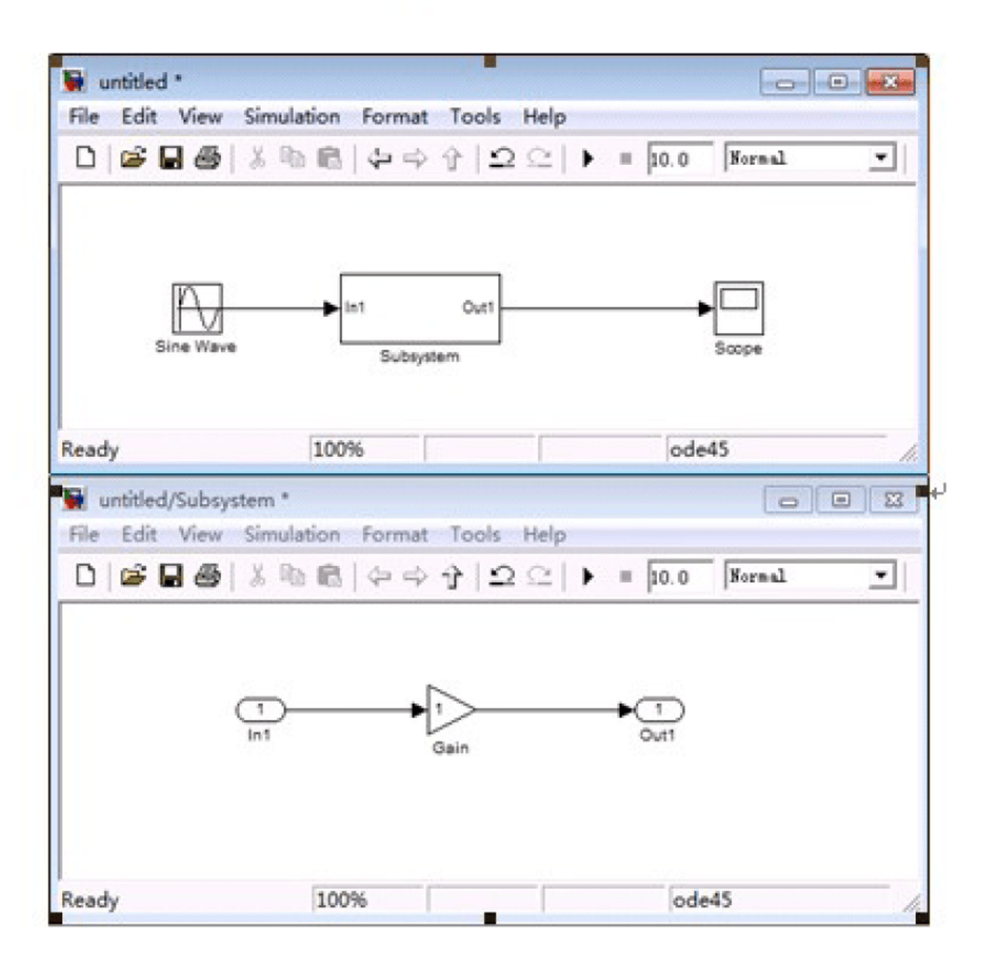


图 6-28　子系统创建实例

6.5.2　子系统的封装

子系统虽然可以使模型更简洁，但是更改子系统内部模块参数时，需要打开许多参数对话框，修改大量的参数时工作会变得十分繁琐，Simulink 提供了封装技术来解决这一问题。封装就是为子系统定制对话框和图标，使子系统具有良好的用户界面。

封装后的子系统与 Simulink 提供的模块一样拥有图标，并且双击图标时会出现一个用户自定义的“参数设置”对话框，实现在对话框中设置子系统中的参数。

1．封装子系统的优点

（1）在设置子系统内部模块参数时可以通过一个参数对话框完成。

（2）为子系统创建一个可以反映子系统功能的图标。

（3）可以避免用户在无意中修改子系统中的模块参数。

2．封装的作用

（1）子系统中各模块的参数通过参数对话框就可以进行设置。

（2）为子系统创建可以反映子系统功能的图标。

（3）可以避免用户在无意中修改子系统中模块的参数。

3．封装的过程

（1）选择需要封装的子系统。

（2）选择【Edit】/【Edit mask】菜单命令，弹出如图 6-29 所示的封装编辑器，设置 Icon、Parameters、Initialization 和 Documentation。

（3）单击【Apply】或【OK】按钮保存设置。

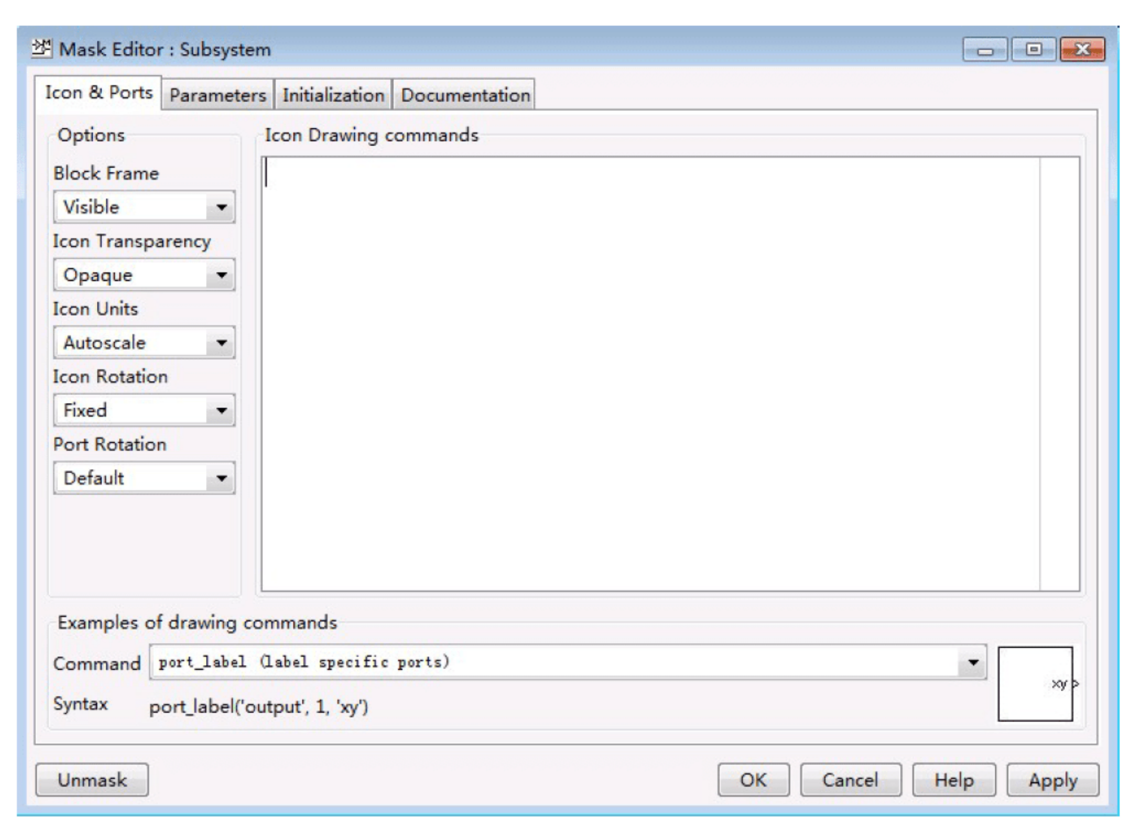


图 6-29　封装编辑器

4．封装编辑器的设置

下面实例将创建一个子系统，并介绍封装编辑器的设置。

【例 6-4】 创建一个子系统，实现斜截式直线方程模型 y=kx+b。

分析：由直线方程可知，该子系统后有两个端口 x 和 y，可以用端口和子系统子库（Ports&Subsysterms）中的输入模块 In1 和输出模块 Out1 表示。子系统本身有两个变量 k 和 b，由方程中的 kx 可以看出，k 可由增益模块 Gain 实现，而 b 则可由常数模块 Constant 实现。

（1）选取模块，在模型窗口中创建模型如图 6-30 所示，创建过程这里不再重复。

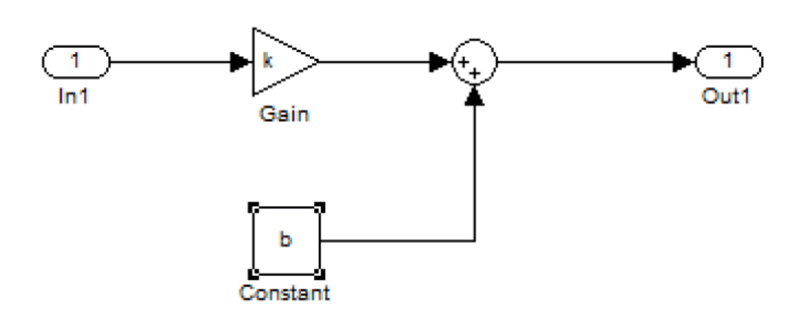


图 6-30 直接创建子系统

（2）设置子系统内模块参数，这里将增益模块参数设置为 k，将常数模块参数设置为 b，如图 6-31 所示。

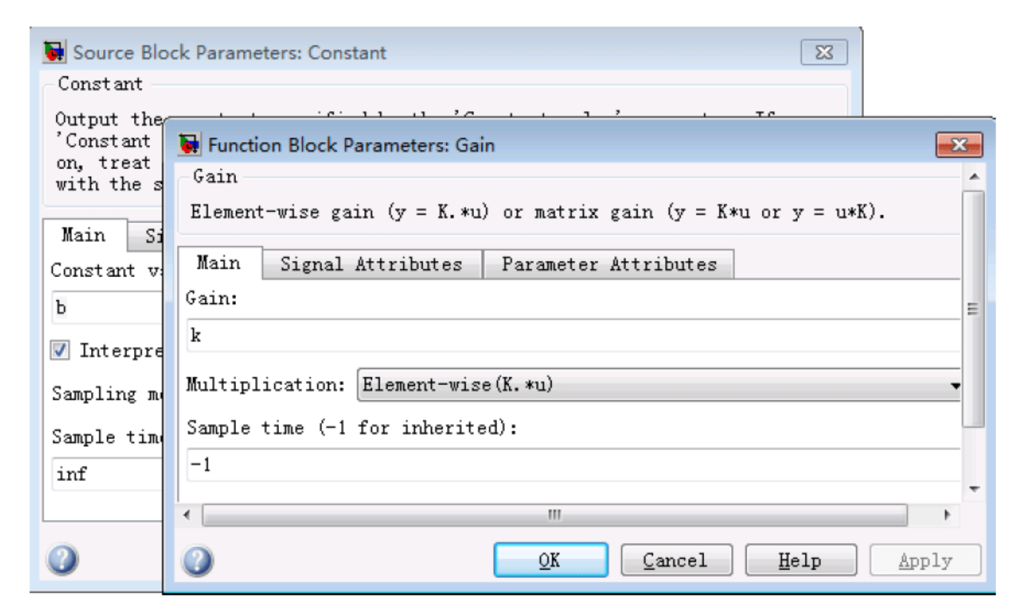


图 6-31　模块参数设置

（3）使用虚线框将要装入子系统部分选中，包括模块和信号线，如图 6-32 所示。

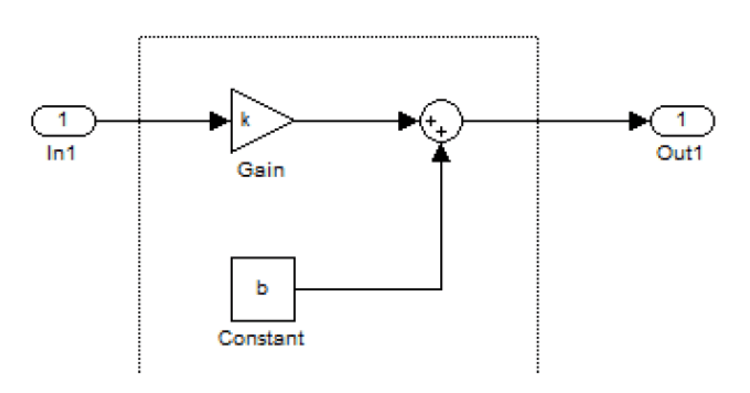


图 6-32　选择创建子系统的对象

（4）选择模型窗口菜单【Edit】/【Create Subsystem】命令，Simulink 将会用一个子系统模块代替所选中的是模块组。适当调整系统模型，最终如图 6-33 所示。

（5）查看如图 6-33 所示的 Subsystem 子系统最简洁的办法就是双击模块，就能看到子系统模块的内部结构图，如图 6-34 所示。

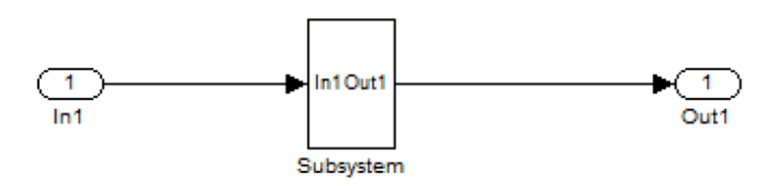


图 6-33　带有子系统的模型

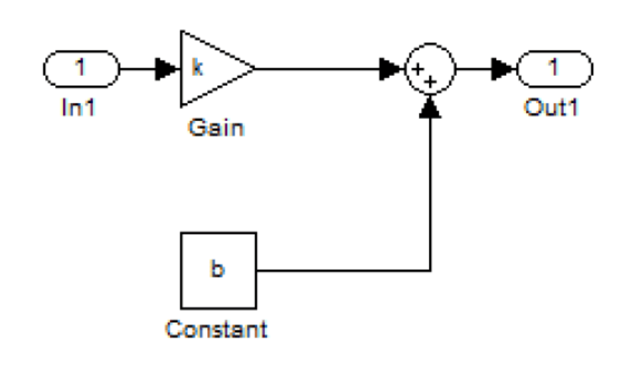


图 6-34　子系统模块内部结构图

【例 6-5】封装在【例 6-4】中构建的子系统模块。

（1）创建子系统，与【例 6-4】相同，最终构建如图 6-35 所示的模型。

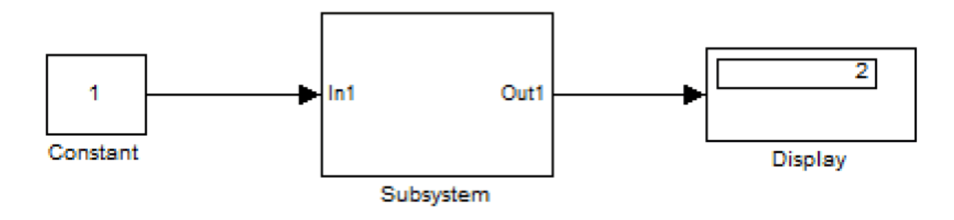


图 6-35　子系统封装模型

（2）设置子系统中模块参数变量。这一步主要是将子系统模块中需要设定的参数在子系统模块内部变量化。本例中，需要对子系统中 Gain 和 Constant 模块中的参数 Gain 和 Constant Value 进行变量化赋值，假设 Gain 设定为变量 k，Constant Value 为 b，设定后子系统内部模型如图 6-36 所示。

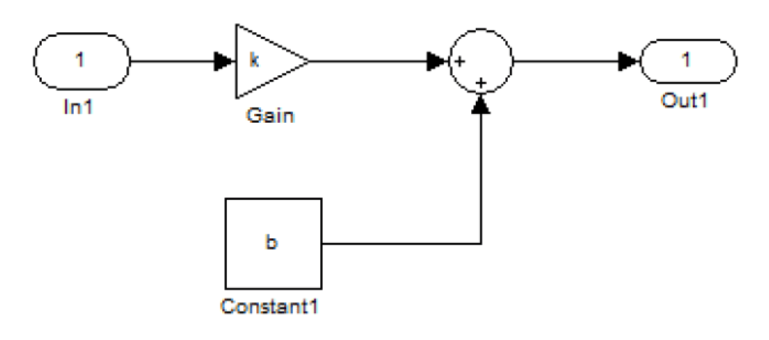


图 6-36　子系统中模块参数变量化

（3）选择子系统模块 Subsystem，选择模型窗口菜单栏中【Edit】/【Mask Subsystem】命令打开封装编辑器，如图 6-37 所示。

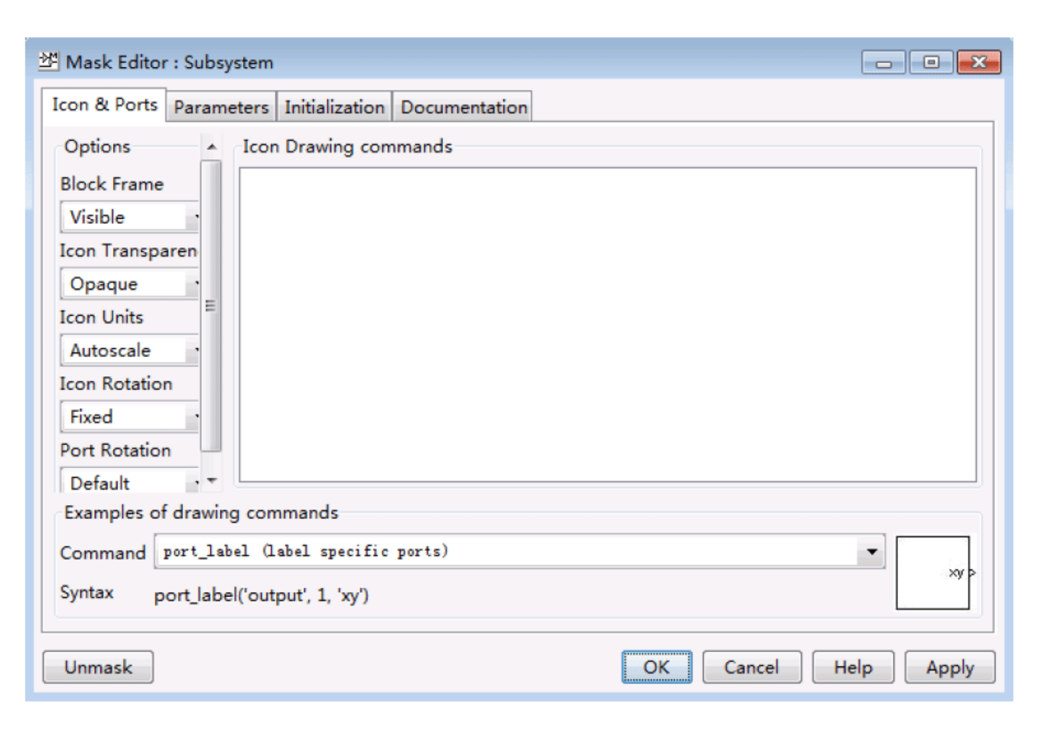


图 6-37　子系统封装编辑器

（4）设置参数标签页 Parameters，该设置主要是为了让子系统模块能够像其他 Simulink 模块一样，具有参数设置对话框，可以设置自己的参数。

- 单击参数设置控制按钮，该键右侧栏将被激活，在其中最上方出现一行空行。
- 在 Prompt 栏中写入“斜率（Slope）”，在 Variable 栏中写入变量名 k；在 Type 栏选择默认的 Edit 类型，表示封装后子系统参数设置界面中 Slope 的变量值通过文本框输入；选择 Evaluate 栏，表示输入量是“数值类”的数值或结果为数值的表达式。
- 再单击参数设置控制按钮，参照前面的方法写入新的一行。按照相同的方式，设置子系统所需的全部参数设置，设置结果如图 6-38 所示。

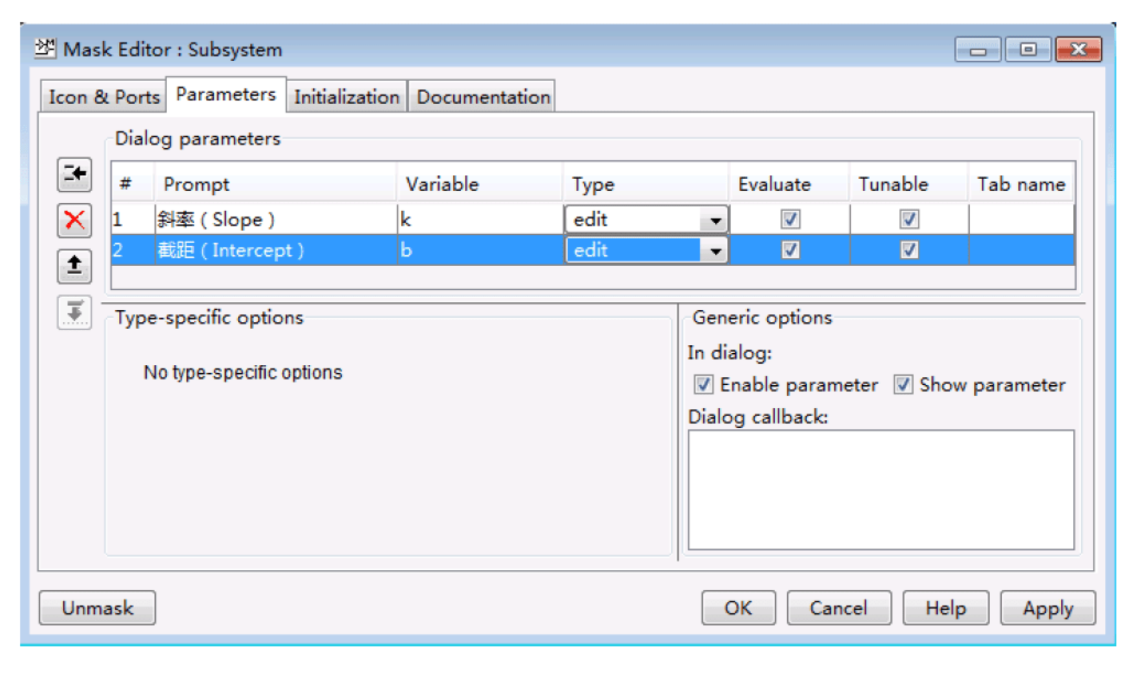


图 6-38　参数标签页设置

（5）设置图标页 Icon。

在 Drawing commands 栏中写入如图 6-39 所示的绘制指令，设置 Transparency 选项为不透明设置。

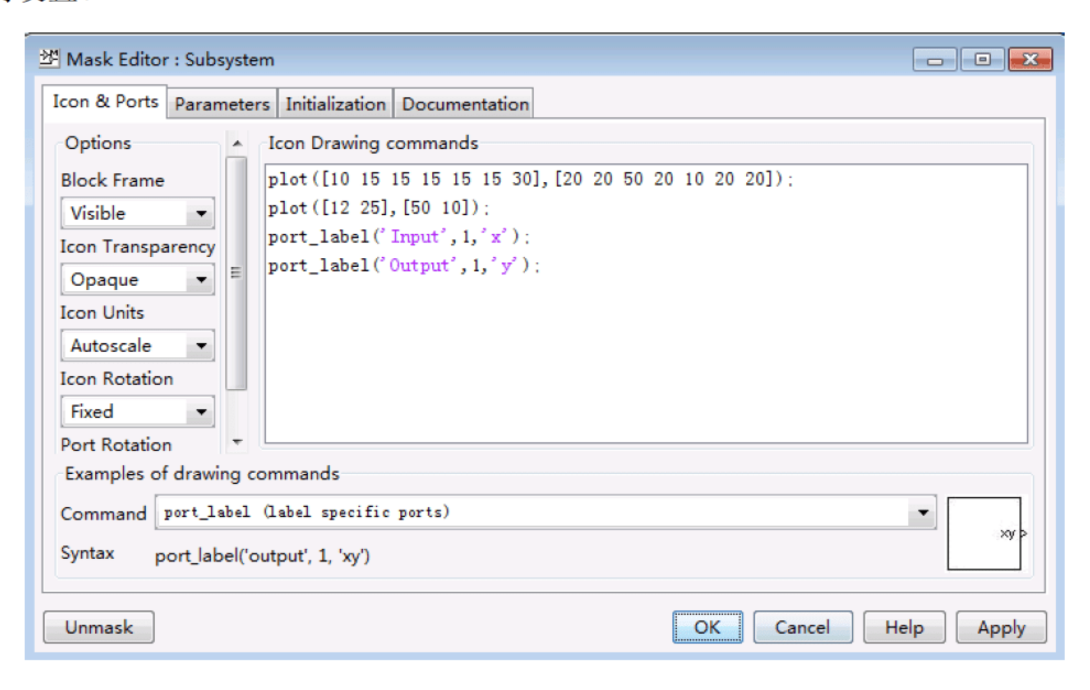


图 6-39　图标标签页设置

（6）设置文档标签页 Documentation。

- ❑ 在 Mask type 栏中输入“斜截式直线方程模块”。
- ❑ 在 Mask description 栏中输入“斜截式直线方程模块，斜率（Slope）和截距（Intercept）是该模块的参数”。
- ❑ 在 Mask help 栏中输入“变量 k 表示斜率，变量 b 表示截距”，设置界面如图 6-40 所示。

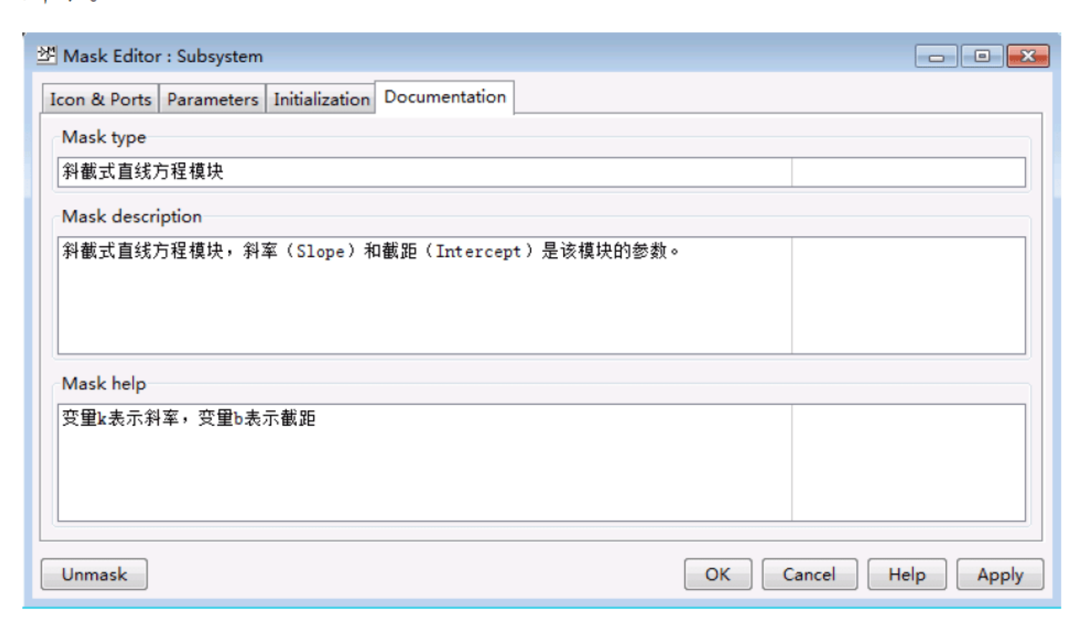


图 6-40　文档标签页设置界面

（7）运行仿真，查看封装结果。

以上基本完成了对子系统的封装，在单击封装编辑器【OK】按钮后将看到如图 6-41 所示的模型图。

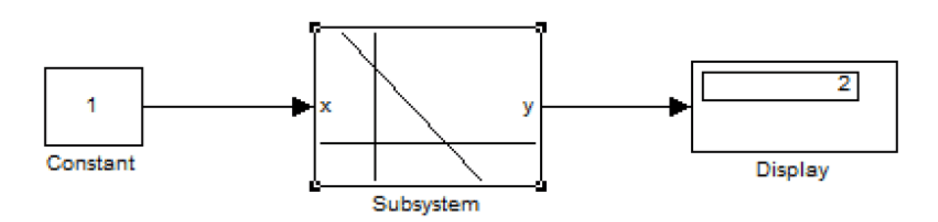


图 6-41　封装后的子系统模型图

双击封装子系统 Subsystem，可以弹出如图 6-42 所示的封装子系统参数设置对话框。

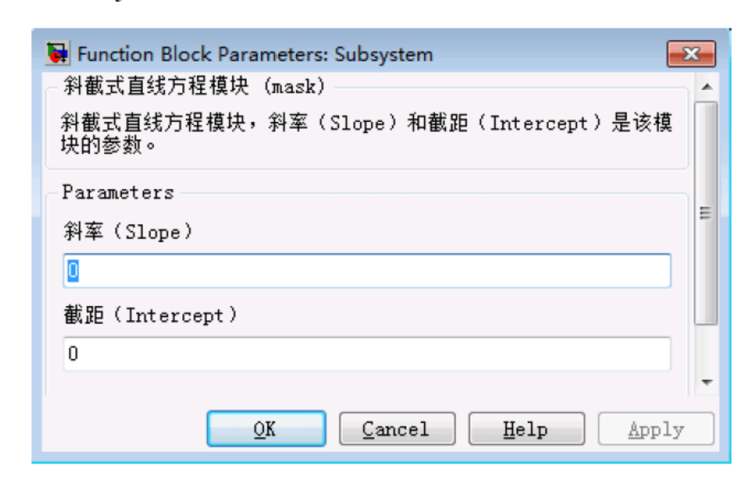


图 6-42　封装后子系统的参数设置对话框

改变封装子系统参数设置对话框中参数斜率 Slope 和截距 Intercept 的值，设置 Slope 为 3，Intercept 为 4，相当于赋值操作，令 k=3、b=4，将所赋值传递给子系统内的模块。此时整个模型实现等式 y=3x+4 的计算。在 Display 中可以实时看到输出值的变化。重新运行仿真后，模型的输出变化如图 6-43 所示。

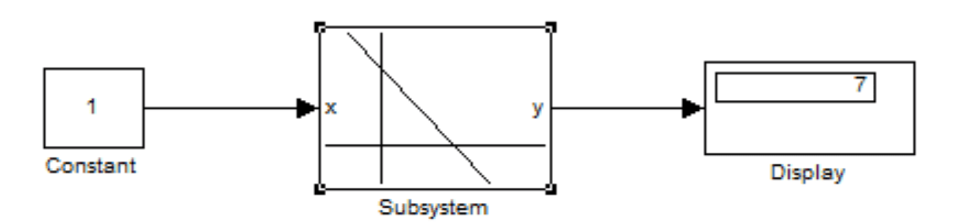


图 6-43　参数调整后的输出变化

6.6　运行仿真

建立模型之后需要运行仿真模型，本节将详细介绍各种仿真参数的设置和仿真的运

行。在介绍仿真运行之前先介绍两个重要的概念：一个是过零检测；另一个是代数环。

6.6.1 过零检测和代数环

1. 过零检测

过零检测通过 Simulink 为模块注册若干过零函数，当变化趋势剧烈时，过零函数发生符号变化。

每个采样点仿真结束时，Simulink 检测是否有过零函数符号变化，如果检测到过零点，Simulink 将在前一个采样点和目前采样点间内插值。

表 6-14 列出了 Simulink 中支持过零检测的模块。

表 6-14 支持过零检测的模块

模　块　名	说　　明
Abs	一个过零检测：检测输入信号沿上升或下降方向通过零点
Backlash	两个过零检测：一个检测是否超过上限阈值；另一个检测是否超过下限阈值
Dead Zone	两个过零检测：一个检测何时进入死区，另一个检测何时离开死区
Hit Crossing	一个过零检测：检测输入何时通过阈值
Integrator	若提供了 Reset 端口，就检测何时发生 Reset；若输出有限，则有 3 个过零检测，即检测何时达到上限饱和值、检测何时达到下限饱和值和检测何时离开饱和区
MinMax	一个过零检测：对于输出向量的每一个元素，检测一个输入何时成为最大或最小值
Rely	一个过零检：若 relay 是 off 状态，就检测开启点：若是 on 状态，就检测关闭点
Relational Operator	一个过零检测：检测输出何时发生改变
Saturation	两个过零检测：一个检测何时达到或离开上限，另一个检测何时达到或离开下限
Sign	一个过零检测：检测输入何时通过零点
Step	一个过零检测：检测阶跃发生时间
Switch	一个过零检测：检测开关条件是满足
Subsystem	用于有条件地运行子系统：一个使能端口，另一个触发端口

2. 代数环

如果 Simulink 模块的输入是依赖于该模块的输出，就会产生一个代数环，如图 6-44 所示。

这意味着无法进行仿真，因为没有输入就得不到输出，没有输出也得不到输入。

解决代数环的办法包括以下几种。

- ❑ 尽量不形成代数环的结构，采用替代结构。
- ❑ 为可以设置初始值的模块设置初值。
- ❑ 对于离散系统，在模块的输出一侧增加 unit delay 模块。
- ❑ 对于连续系统，在模块的输出一侧增加 memory 模块。

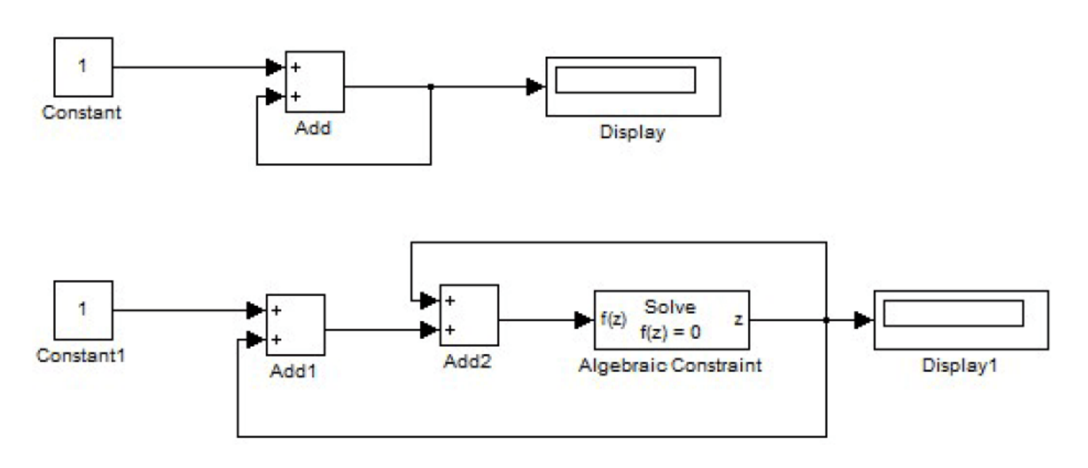


图 6-44　代数环样例

6.6.2　仿真的运行

1. 使用窗口运行仿真

建立好模型后，可以通过选择【Simulink】/【Start】菜单命令进行仿真，或如图 6-45 所示通过单击工具栏上的开始按钮进行仿真。

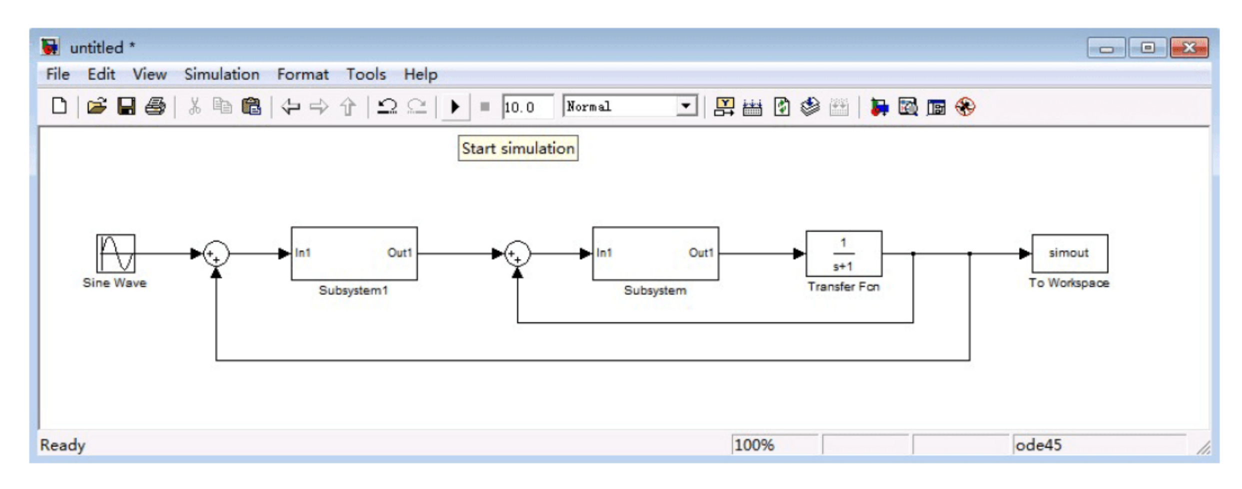


图 6-45　通过工具栏进行仿真

2. 使用 MATLAB 命令运行仿真

MATLAB 允许通过命令窗口运行仿真。MATLAB 提供函数 sim()运行仿真，其具体使用方法如下。

- [t,x,y]=sim(filename,timespan,options,ut);
- [t,x,y1,y2,…..yn]=sim(filename,timespan,options,ut);

只有参量 filename 是必须的，各参量的含义如表 6-15 所示。

表 6-15　函数 sim()参量

参　量　名	参 量 含 义
T	返回仿真时间
X	返回仿真的状态矩阵

续表

参 量 名	参 量 含 义
Y	返回仿真输出矩阵
Y1，Y2，...Yn	每一个 Yi 对应一个输出模块
Filename	字符串类型，并且模型保存为 filename
Timespan	设置仿真的开始和结束时间
Options	用于设置仿真相关参数的一个结果
Ut	模型输入

6.7 模型调试

Simulink 提供了调试器，以方便查找和诊断模型中的错误，它允许通过单步运行仿真显示模块的即时状态、输入和输出。

1．Simulink 调试器

工具栏按钮及功能介绍如表 6-16 所示。

表 6-16 工具栏功能介绍

工具栏按钮	功 能	工具栏按钮	功 能
	进入当前方法		停止仿真
	跳过当前方法		运行到下一个模块前跳出
	跳出当前方法		当选中的模块被执行时显示其输入输出
	在下一个仿真时间步跳转到第一个方法		显示选中的模块的当前输入输出
	跳转到下一个模块方法		选择动画模式
	开始或继续调试	?	显示调试器的帮助
	暂停仿真	Close	关闭调试器

2．命令行调试

许多 Simulink 命令和消息是通过 Method ID 和 Block ID 来引用方法和模块的。

❑ Method ID 是按方法被调用的顺序从 0 开始分配的一个整数。

❑ Block ID 在编译阶段分配，形式为 sid:bid。

3．设置断点

断点就是使仿真运行到该位置时停止，同时可以使用命令 continue 使仿真继续运行。调试器允许定义无条件断点和有条件断点。

1）设置无条件断点

设置无条件断点有如下 3 种方式。

❑ 通过调试器工具栏；

❑ 通过调试器 Simulation Loop 页；

- ❑ 通过在 MATLAB 命令窗口运行相关命令。

2）设置有条件断点

设置有条件断点可以通过在调试器“Break on conditions”页中设置相应的断点条件来实现。

4．显示仿真的信息

Simulink 调试器工具条中的按钮用于显示模块的输入/输出信息。

- ❑ 首先在模型窗口选中模块。
- ❑ 然后单击该按钮，被选中的模块在当前采样点的输入、输出和状态信息将显示在调试器窗口的“Outputs”页中。

5．显示模型的信息

调试器除了可以显示仿真的相关信息外，还可以显示模型的相关信息。

在 MATLAB 命令窗口中，可以用命令 slist 显示系统中各模块的索引，模块的索引就是它们的执行顺序，它与调试器窗口中“Sorted List”页显示的内容相同。

6.8　S–函数

Simulink 为用户提供了许多内置的基本库模块，通过这些模块进行连接而构成系统的模型。对于那些经常使用的模块进行组合并封装可以构建出重复使用的新模块，但它依然是基于 Simulink 原来提供的内置模块。

S-函数是一个动态系统的计算机语言描述，在 MATLAB 里，用户可以选择用 M 文件编写，也可以用 C 语言、C++语言、Ada 或 Fortran 语言编写，这些语言编写的 S-函数被编译成 MEX-文件，在需要时，被连接到 MATALAB。本节主要介绍如何用 M 文件编写 S-函数。

S-函数提供了扩展 Simulink 模块库的有力工具，它采用一种特定的调用语法，使函数和 Simulink 解法器进行交互。

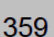
S-函数最广泛的用途是定制用户自己的 Simulink 模块，其形式通用，能够支持连续系统、离散系统和混合系统。

6.8.1　S–函数的概念

在 S-函数的编写中会遇到下面一些基本概念，如直接反馈（Direct feedthrough）、动态输入（Dynamically sized inputs）、设置采样时间和偏移（Setting sample times and offsets）。理解这些概念对于正确创建 S-函数是非常重要的。

1．直接反馈（Direct feedthrough）

直接反馈，是指系统的输出或可变采样时间受到输入的控制。简单地说，就是如果输出信号是输入信号的函数，或者在可变步长仿真过程中，S-函数影响着下一个仿真时刻的计算，那么就是直接反馈。有些系统具有直接反馈性，而有些没有。如系统 y=ku（u 是输入，k 是增益系数，y 是输出），就具有直接反馈性。而系统 y=x，dx=u，x 表示状态，就



不具有直接反馈性。

要确定模块的执行顺序，就需要判断 S-函数有无直接反馈性。一般来说，判断 S-函数输入端口是否总具有直接反馈性的依据如下。

- 从 S-函数的角度看，输出函数中包含有输入 u 的函数。
- 下一采样时刻的计算需要输入 u。

2. 动态输入（Dynamically sized inputs）

S-函数可以动态设置输入向量宽度（维数）。S-函数的输入变量的宽度取决于 S-函数输入模块的宽度。动态输入主要是给出输入连续状态数目（Size.NumContStates）、离散状态数目（Size.NumDiscStates）、输出数目（Size.NumOutputs）、输入数目（Size.NumInputs）和直接反馈数目（Size.Dir Feedthrough）。

S-函数只有一个输入输出端口，所以其只能接受一维输入向量。动态设置输入向量宽度时，可以将指定 Size 结构的对应成员设置为−1，也可以在仿真开始时，调用 length 函数来确定实际输入向量的宽度。若指定宽度为 0，则对应的输入端口将会在 S-函数模块中去掉。

3. 设置采样时间和偏移（Setting sample times and offsets）

设置采样时间和偏移设置主要设置采样时间。M 文件 S-函数和 C 语言 S-函数都具备在指定 S-函数的执行时间上有高度的自适应度。Simulink 为采样时间提供了以下不同的选择。

- 连续采样时间（**Continuous sample time**） 适用于具有连续状态和非采样过零点的 S-函数，其输出按照最小时间步改变。
- 固定最小步长的连续采样时间（**Continuous but fixed in minor time step sample time**） 适用于需要在每一个主仿真时间步执行，但在最小仿真步内值不改变的 S-函数。
- 离散采样时间（**Discrete sample time**） 在 S-函数发生了具有离散时间间隔的函数行为，用户可以定义一个采样时间来规定 Simulink 何时调用函数。而且用户还可以定义一个延迟时间 offset 来延迟采样时间，这个延迟时间不能超过采样时间。若用户定义了一个离散采样时间，则 Simulink 就会在所定义的每个采样点调用 S-函数的 mdlOutput 例程和 mdlUpdate 例程。
- 可变采样时间（**Variable sample time**） 相邻采样点的时间间隔可变的离散采样时间。在这种采样时间的情况下，S-函数将会在下一步仿真开始时，计算下一个采样点的时刻。
- 继承采样时间（**Inherited sample time**） 在某些情况下，S-函数自身没有特定的采样时间，它本身的状态是连续还是离散的完全取决于系统中的其他模块。此时，S-函数模块的采样时间属性可设置为继承（inherited）

通常，一个模块可以从以下方式中继承采样时间：继承驱动模块（The driving block）；继承目标模块（The destination block）；系统中最快的采样时间。

- 在最小积分步长内会发生变化的连续 S 函数，应将采样时间设置为

```
[continous_sample_time,0.0 ]
```

- 在最小积分步长内不会发生变化的连续 S 函数，应将采样时间设置为

```
[continous_sample_time,fixed_in_minor_step,offset ]
```

❑ 以固定速率变化的离散 S 函数，应将采样时间设置为

```
[discrete_sample_time_period,offset ]
```

❑ 变速率的离散 S 函数，应将可变步长的离散采样时间设置为

```
[variable_sample_time,0.0 ]
```

6.8.2 S-函数的工作原理

要创建一个 S-函数，了解 S-函数的工作原理就显得尤为重要。S-函数的一个优点就是可以创建一个通用的模块，在模型中可以多次调用，在不同的场合下仅仅修改它的参数就可以了。因此，在了解 S-函数的工作原理之前，首先了解模块的共同特性，以便能够更好地理解 Simulink 的整个仿真原理，然后将简介 Simulink 的仿真阶段和 S-函数的反复调用。

1. Simulink 模块的共同特性

Simulink 模块包含 3 个基本元素：输入向量（u）；状态向量（x）；输出向量（y）。如图 6-46 所示，显示了 Simulink 模块 3 个基本单元的关系。

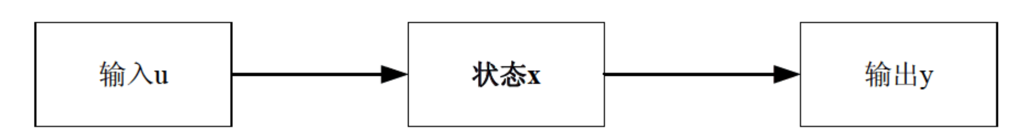


图 6-46　模块的输入、输出、和状态关系图

输入、状态和输出之间的数学关系可用状态方程描述为 $y=f_0(t,x,u)$；$x_c= f_d(t,x,u)$；$x_d= f_u(t,x,u)$ 其中 $x= x_c+x_d$。

2. Simulink 仿真阶段

Simulink 的仿真阶段分为两个阶段：第一个阶段为初始化阶段，在这个阶段，模块的所有参数将传递给 MATLAB 进行计算，所有参数将被确定下来，同时，Simulink 将展开模型的层次，每个子系统被它们所包含的模块替代，传递信号宽度、数据类型和采样时间，确定模块的执行顺序，最后确定模块的初值和采样时间。第二个阶段是仿真阶段，在这个阶段主要进行模块输出的计算，更新模块的离散状态，计算连续状态，在采用变步长解法器时还需要确定时间步长。

3. S-函数的反复调用

Simulink 模型中反复调用 S-函数，以便执行每一阶段的任务。Simulink 会对模型中的 S-函数采用适当的方法进行调用，在调用过程中，Simulink 将调用 S-函数来完成以下各项任务。

（1）初始化：在仿真开始前，Simulink 在这个阶段初始化 S-函数，这些工作包括以下几项。

❑ 初始化结构体 SimStruct，它包含了 S-函数的所有信息。

❑ 设置输入输出端口的数目和大小。

❑ 设置采样时间。

❑ 分配存储空间并估计数组大小。

（2）计算下一个采样时间点：如果选择步长解法器进行仿真时，需要计算下一个采样

时间点，即计算下一步的仿真步长。

（3）计算主要时间步的输出：计算所有端口的输出值。

（4）更新状态：此例程在每个步长处都要执行一次，可以在这个例程中添加每一个仿真步都需要更新的内容，如离散状态的更新。

（5）数值积分：用于连续状态的求解和非采样过零点。如果 S-函数存在连续状态，Simulink 就在 minor step time 内调用 mdlDdrivatives 和 mdlOutput 两个 S-函数例程。

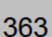
6.8.3 S–函数模板

Simulink 中为用户编写 S-函数提供了多种模板文件，该模板文件定义了完整的 S-函数框架结构，用户可以根据自己的需要来修改模板。编写 M 文件 S-函数时，推荐使用 S-函数模板文件，即 sfuntmpl.m，该文件存储在 MATLAB 的根目录 toolbox\simulink\blocks 子目录中。sfuntmple.m 模板文件是一个 M 文件 S-函数，由一个主函数和六个子函数组成，在主函数程序内根据标志变量 Flag，由一个 Switch-Case 语句根据标志值将 Simulink 转移到相应的子函数中。

下面给出 sfuntmpl.m 模板文件源代码，其中删除了原有的注释代码，为了读者便于理解，本书在适当的位置添加了中文注释。

```
function [sys,x0,str,ts] = sfuntmpl(t,x,u,flag)
%x0 是状态变量的初始值。
%一般在初始化中将 str 置空就可以了
%ts 是一个 1*2 的向量，ts(1)是采样周期，ts(2)是偏移量
%函数名 sfuntmpl 是模板文件名，用户在编辑时应编写自己的文件名
% t 是采样时间，x 是状态变量，u 是输入
%flag 是仿真过程中的状态标志，它的 6 个不同的权值分别指向 6 个功能不同的子函数。
%这些子函数也称为回调方法。
%sys 输出根据 flag 的不同而不同，下面将结合 flag 来讲 sys 的含义
switch flag,
%判断 flag，看当前处于哪个状态
  case 0,
[sys,x0,str,ts]=mdlInitializeSizes;
%调用"模块初始化"子函数
  case 1,
sys=mdlDerivatives(t,x,u);
%调用"计算模块导数"子函数
  case 2,
sys=mdlUpdate(t,x,u);
%调用"更新模块离散状态"子函数
  case 3,
sys=mdlOutputs(t,x,u,k);
%调用"计算模块输出"子函数
  case 4,
sys=mdlGetTimeOfNextVarHit(t,x,u);
%调用“计算下一个采样时间点”子函数
  case 9,
sys=mdlTerminate(t,x,u);
```

```
%调用"结束仿真"子函数
  otherwise
    error(['Unhandled flag = ' ,num2str(flag)]);
end
%=================================================================
function [sys,x0,str,ts]=mdlInitializeSizes
%模块初始化子函数
sizes = simsizes;
%调用 simsizeS-函数，返回规范的 Sizes 构架，这个指令用户无须改动
sizes.NumContStates  = 0;
%模块连续状态的数目。这里 0 是模板的默认值，用户可以根据自己所描述的系统进行修改
sizes.NumDiscStates  = 0;
%模块离散状态的数目。这里 0 是模板的默认值，用户可以根据自己所描述的系统进行修改
sizes.NumOutputs     = 0;
%模块输出的数目。这里 0 是模板的默认值，用户可以根据自己所描述的系统进行修改
sizes.NumInputs      = 0;
%模块输入的数目。这里 0 是模板的默认值，用户可以根据自己所描述的系统进行修改
sizes.DirFeedthrough = 1;
%模块是否存在直接馈入。有则置为 1，无则置为 0，这里 1 是模板的缺省值
sizes.NumSampleTimes = 1;
%模块的采样时间个数，至少是一个。用户可根据自己所描述的系统进行修改
sys = simsizes(sizes);
%初始完后 sizes 向 sys 赋值
x0  = [];
%设置初始状态，缺省为空
str = [];
%保留参数，缺省为空，用户不必修改
ts  = [0 0];
%设置采样时间和偏移量
%=================================================================
function sys=mdlDerivatives(t,x,u)
%计算模块导数子函数。在此处填写计算导数向量的指令
sys = [];
%用户必须把算得的导数向量赋给 sys，这里的[ ]是默认设置
%=================================================================
function sys=mdlUpdate(t,x,u)
%更新模块离散状态子函数。在此处填写计算离散状态向量的指令
sys = [];
%用户必须把算得的离散状态向量赋给 sys，这里的[ ]是默认设置
%=================================================================
function sys=mdlOutputs(t,x,u,k)
%计算模块输出子函数。在此处填写计算模块输出向量的指令
sys = [];
%用户必须把算得的模块输出向量赋给 sys，这里的[ ]是默认设置
%=================================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u)
%计算下一个采样时间点子函数。该子函数只有在"变采样时间"下使用
sampleTime = 1;
%表示在当前时刻 1 秒后再调用本模块。用户可根据需要修改
sys = t + sampleTime;
%将算得的下一采样时刻赋给 sys。用户无须修改
```

```
%==============================================================
function sys=mdlTerminate(t,x,u)
%结束仿真子函数。
sys = [];
%系统默认为[ ]，一般不需改动
```

6.8.4 S–函数的使用

在 Simulink 中，User Defined Function 子库中有一个 S-Function 模块，用户可以利用该模块在模型中创建 S-函数。一般来说，Simulink 可以通过如下步骤来实现创建包含 S-函数的模型。

（1）打开 Simulink 库浏览器，将 User Defined Function 子库中 S-Function 模块复制到用户模型窗口中。

（2）双击 S-Function 模块，打开其参数设置对话框，如图 6-47 所示，设置 S-函数参数。在 S-函数文件名区域要填写 S-函数不带扩展名的文件名，在 S-函数参数编辑框中填入 S-函数所需要的参数，参数并列给出，参数间以逗号隔开。

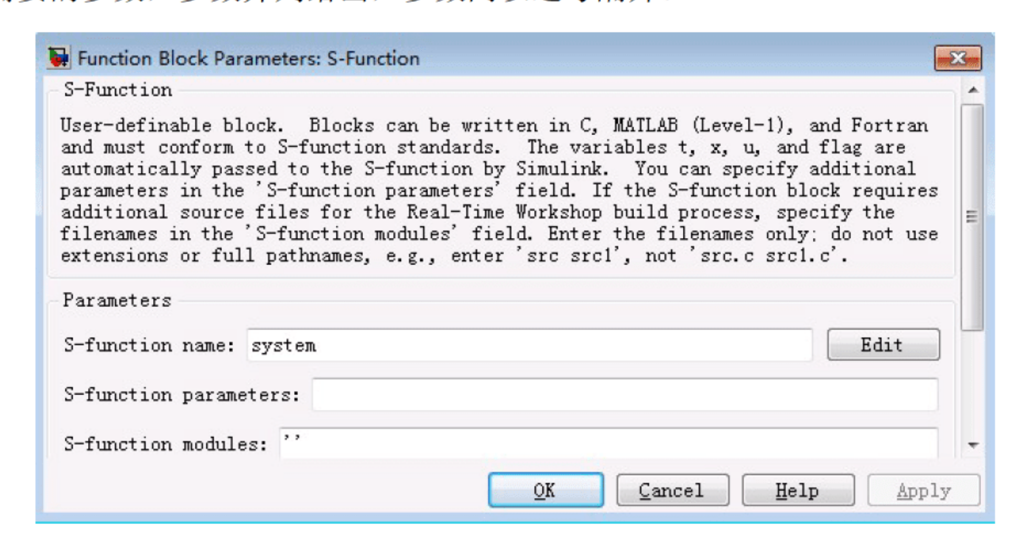


图 6-47　S-函数模块参数对话框

（3）创建 S-函数源代码，单击 S-函数模块参数对话框中的 Edit ，即可打开源代码编辑窗口，如图 6-48 所示。事实上，S-函数源代码的创建方式有很多种，一般来说，在 Simulink 的 S-function Example 模型库中，Simulink 为用户提供了针对不同语言的 S-函数模板和实例。用户通过修改 S-函数的模板和例子来实现 S-函数源文件编写工作，然后直接在 S-函数模块参数对话框中输入已经编辑好的 S-函数名，即可直接调用。

（4）在 Simulink 仿真模型中，连接模块，进行仿真。

需要注意的是，用户可以利用子系统封装功能对 S-函数进行封装，以提供更加友好的使用界面；S-Function 参数设置中的 S-函数文件名必须与用户建立的 S-函数源文件名完全相同；用户必须知道 S-函数要求的参数和这些参数的调用顺序，然后按照 S-函数的要求的顺序输入参数；S-函数是一个单输入、单输出的模块，如果系统有多个输入或输出信号，

则需要使用 Mux 和 Demux 模块将其组合成单个的输入或输出信号。

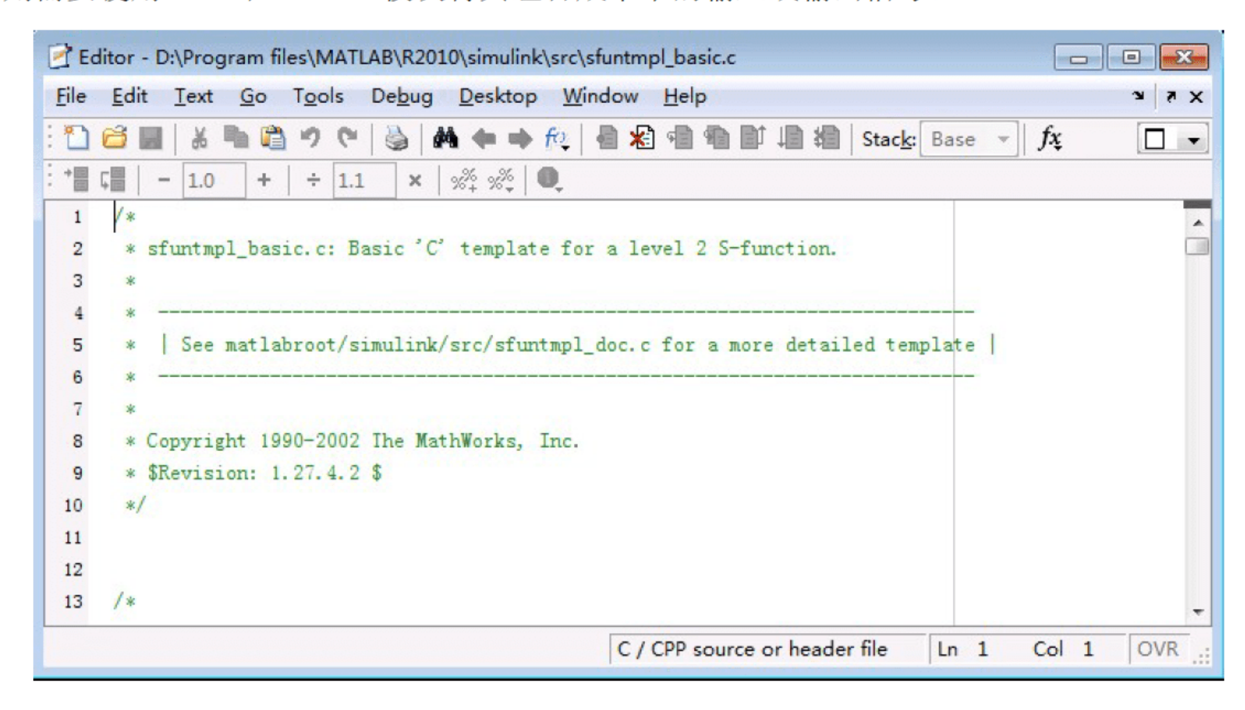


图 6-48　源代码编辑窗口

下面结合具体实例来说明一下 S-函数的使用。

【例 6-6】 使用 S-函数实现【例 6-4】的仿真，即利用 S-函数实现斜截式直线方程模块。

（1）构建模型。构建如图 6-49 所示的模型。

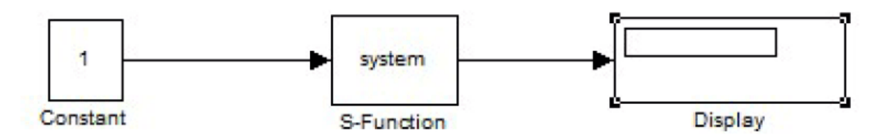


图 6-49　S-函数使用演示模型

（2）打开标准模板文件 sfuntmpl。打开模板文件，有三种方式可以将其打开。

- 在 MATLAB 命令窗口中输入>> open sfuntmpl.m。
- 在 MATLAB 命令窗口中输入>> edit sfuntmpl。
- 在 Simulink 库浏览器中，双击 User-defined Function\S-Function Examples\MATLAB file S-functions\Leveal-1 MATLAB file S-functions \ Leveal-1 MATLAB file template 模块。

（3）修改模板文件，完成 S-函数源代码编写。修改后的完整代码如下，所需修改的代码处已经标注出。

```
function [sys,x0,str,ts] = Sfun_line(t,x,u,flag,k,b)
%在主函数中修改函数名称，输入 S-函数模块需要设置的参数 k 和 b
switch flag,
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
  case 1,
```

```
    sys=mdlDerivatives(t,x,u);
  case 2,
    sys=mdlUpdate(t,x,u);
  case 3,
    sys=mdlOutputs(t,x,u);
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
  case 9,
    sys=mdlTerminate(t,x,u);
  otherwise
    error(['Unhandled flag = ' ,num2str(flag)]);
end
%==================================================================
function [sys,x0,str,ts]=mdlInitializeSizes
%初始化：在 mdlInitializeSizes 中，确定输入和输出数目
%对于带有至少一个输出和输入的简单系统，它总是直接反馈的
sizes = simsizes;

sizes.NumContStates  = 0;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 1;
sizes.NumInputs      = 1;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;   % at least one sample time is needed
sys = simsizes(sizes);
x0  = [];
str = [];
ts  = [0 0];
%==================================================================
function sys=mdlDerivatives(t,x,u)
sys = [];
%==================================================================
function sys=mdlUpdate(t,x,u)
sys = [];
%==================================================================
%在 mdlOutputs 函数中，编写输出方程，并通过变量 sys 返回
function sys=mdlOutputs(t,x,u,k,b)
sys = [k*u+b];
%==================================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;
sys = t + sampleTime;
%==================================================================
function sys=mdlTerminate(t,x,u)
sys = [];
```

（4）双击 S-Function 模块，打开模块参数设置对话框，如图 6-50 所示。

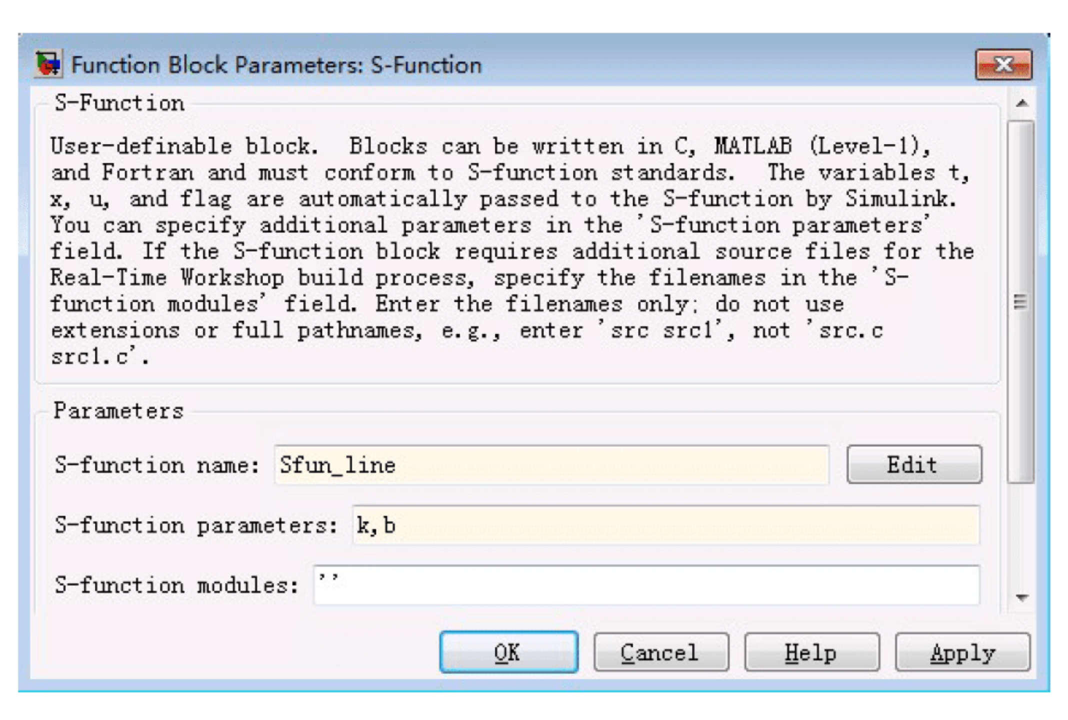


图 6-50　设置模型参数对话框

（5）封装 S-函数模块，这里的封装步骤和【例 6-5】的步骤完全相同，请参照【例 6-5】，封装后的导入模型如图 6-51 所示。

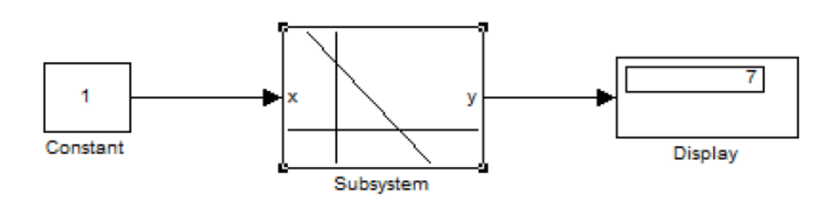


图 6-51　封装后的 S-函数模块

（6）设置 S-Function 模块参数对话框，并进行仿真，这里的步骤与【例 6-5】的步骤完全相同，不再重复。

6.8.5　S–函数举例

在本小节中，将通过修改标准模板，向读者介绍一些经典的实例，如含参 S-函数、连续状态系统的 S-函数描述、离散状态系统的 S-函数描述等，通过这些实例向读者介绍 M 文件 S-函数源代码的编写方法。事实上，含参数的 S-函数创建在【例 6-5】中已经介绍，下面主要介绍连续状态 S-函数描述和离散状态 S-函数描述。

【例 6-7】 连续状态 S-函数描述。假设线性连续系统的状态方程为：$\begin{cases} x = Ax + Bu \\ y = Cx + Du \end{cases}$；其中，$A = \begin{bmatrix} -0.09 & -0.01 \\ 1 & 0 \end{bmatrix}$，$B = \begin{bmatrix} 1 & -7 \\ 0 & -2 \end{bmatrix}$，$C = \begin{bmatrix} 0 & 2 \\ 1 & -5 \end{bmatrix}$，$D = \begin{bmatrix} -3 & 0 \\ 1 & 0 \end{bmatrix}$。创建 S-函数描述该系统。

（1）打开 sfuntmpl 模板，该系统可通过在 sfuntmpl 模板编写如下源代码实现。

```
function [sys,x0,str,ts] = sfun c(t,x,u,flag)
%定义连续系统的 S-函数 sfun_c。生成连续系统状态
A=[-0.09  -0.01  ; 1  0];
B=[1  -7   ;  0  -2];
C=[0   2   ;  1  -5];
D=[-3  0   ;  1  0];
switch flag,
%初始化状态
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D);
%计算连续状态变量
  case 1,
    sys=mdlDerivatives(t,x,u, A,B,C,D);
%由于含有状态导数，故 mdlDerivatives 函数调用需要修改
  case 2,
    sys=mdlUpdate(t,x,u);
%计算系统输出
  case 3,
    sys=mdlOutputs(t,x,u,A,B,C,D);
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
  case 9,
    sys=mdlTerminate(t,x,u);
%处理错误
  otherwise
    error(['Unhandled flag = ' ,num2str(flag)]);
end
%==============================================================
function [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D);
sizes = simsizes;

sizes.NumContStates  = 2;
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;
sizes.NumInputs      = 2;
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;   % at least one sample time is needed
sys = simsizes(sizes);
x0  = zeros(2,1);
str = [];
ts  = [0 0];
%==============================================================
function sys=mdlDerivatives(t,x,u,A,B,C,D)
%状态方程含有状态的导数，故需要编写 mdlDerivatives 子函数，将状态的导数通过 sys 变量
返回
sys = A*x+B*u;
%==============================================================
function sys=mdlUpdate(t,x,u)
sys = [];
```

```
%==================================================================
function sys=mdlOutputs(t,x,u,A,B,C,D)
sys =C*x+D*u;
%==================================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;
sys = t + sampleTime;
%==================================================================
function sys=mdlTerminate(t,x,u)
sys = [];
%结束仿真
```

（2）保存该 M 文件 S-函数文件名为 sfun_c.m。

【例 6-8】 创建 S-函数描述该离散系统。

假设离散系统的状态方程为 $\begin{cases} x(k+1)=Ax(k)+Bu(k) \\ y(k)=Cx(k)+Du(k) \end{cases}$ 其中，$A=\begin{bmatrix} -1.3 & -0.5 \\ -1.0 & 0 \end{bmatrix}$，$B=\begin{bmatrix} -2.5 & 0 \\ 0 & 4.3 \end{bmatrix}$，$C=\begin{bmatrix} 0 & 2.1 \\ 1 & 7.8 \end{bmatrix}$，$D=\begin{bmatrix} -0.8 & -2.9 \\ 1.2 & 0 \end{bmatrix}$。

（1）打开 sfuntmpl 模板，该系统可通过在 sfuntmpl 模板编写如下源代码实现。

```
function [sys,x0,str,ts] = sfun_d(t,x,u,flag)
%定义离散系统的 S-函数 sfun d。生成离散系统状态
A=[-1.3  -0.5  ; -1.0  0];
B=[-2.5  0   ;  0  4.3];
C=[0   2.1  ; 1  7.8];
D=[-0.8  -2.9  ; 1.2  0];
switch flag,
%初始化状态
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D);
  case 1,
    sys=mdlDerivatives(t,x,u, A,B,C,D);
%更新离散状态
  case 2,
    sys=mdlUpdate(t,x,u);
%计算系统输出
  case 3,
    sys=mdlOutputs(t,x,u,A,B,C,D);
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
  case 9,
    sys=mdlTerminate(t,x,u);
%处理错误
  otherwise
    error(['Unhandled flag = ' ,num2str(flag)]);
end
%==================================================================
```

```
function [sys,x0,str,ts]=mdlInitializeSizes(A,B,C,D);
sizes = simsizes;
sizes.NumContStates  =0;
sizes.NumDiscStates  = size(A,1);
sizes.NumOutputs     = size(D,1);
sizes.NumInputs      = size(D,1);
sizes.DirFeedthrough = 1;
sizes.NumSampleTimes = 1;   % at least one sample time is needed
sys = simsizes(sizes);
x0  = zeros(sizes.NumDiscStates,1);
str = [];
ts  = [1 0];
%==================================================================
function sys=mdlDerivatives(t,x,u)
sys = [ ];
%==================================================================
%更新离散状态子函数
function sys=mdlUpdate(t,x,u,A,B,C,D)
sys = A*x+B*u;
%==================================================================
%计算输出子函数
function sys=mdlOutputs(t,x,u,A,B,C,D)
sys =C*x+D*u;
%==================================================================
function sys=mdlGetTimeOfNextVarHit(t,x,u)
sampleTime = 1;
sys = t + sampleTime;
%==================================================================
function sys=mdlTerminate(t,x,u)
sys = [];
%结束仿真
```

（2）保存该 M 文件 S-函数文件名为 sfun_d.m。

6.9　本章小结

Simulink 是用于动态系统和嵌入式系统的多领域仿真和基于模型的设计工具。对各类系统，包括通信、控制、信号处理、视频处理和图像处理系统，Simulink 提供了交互式图形化环境和可指定模块库来对其进行设计、仿真、执行和测试。通过本章的学习，掌握如何建模，如何设置仿真参数，如何实现仿真是本章学习的重点；而子系统和 S-函数的编写则是 Simulink 技术的提升，读者可采用 S-函数定制自己的模块库，通过子系统的方式来实现复杂的建模仿真。

6.10 习题

（1）有初始状态为 0 的二阶微分方程 x"+0.2x'+0.4x=0.2u (t)，其中，u(t)是单位阶跃函数，试建立系统模型并仿真。

（2）新建一个 Simulink 的模型文件，试建立并调试一个模型，实现在一个示波器中同时观察正弦波信号和方波信号。

（3）食饵——捕食者模型：设食饵（如鱼，兔等）数量为 x(t)，捕食者（如鲨鱼，狼等）数量为 $y(t)$，有 $\begin{cases} \dot{x} = x(r - ay) \\ \dot{y} = y(-d + bx) \end{cases}$ 或 $\begin{pmatrix} \dot{x} \\ \dot{y} \end{pmatrix} = \begin{pmatrix} r - ay & 0 \\ 0 & -d + bx \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix}$，设 r=1，d=0.5，a=0.1，b=0.02, $x(0)$=25,$y(0)$=2。求 $x(t)$, $y(t)$和 $y(x)$的图形。

第 7 章　图形用户界面

本章主要介绍图形用户界面（Graphical User Interface，GUI）的设计，以便更好地进行人机交互。在 MATLAB 中有两种图形用户界面的设计方法，即纯 M 文件编程的方式和利用 GUIDE（Graphical User Interface Development Environment）的方式。由于利用 GUIDE 的方式在设计过程中更直观，所见即所得，同时减少了编码工作，所以本章着重介绍该方式。

7.1　界面设计

图形用户界面（Graphical User Interface，GUI）设计，就是根据需要设计出由窗口、菜单、按钮及文字说明等对象所构成的图形界面，用户可以通过鼠标或键盘在该界面进行相应操作。在 MATLAB 所提供的示例中有很好的图形用户界面演示程序。

图形用户界面的设计可以通过两种方法来实现，即在可视化界面环境中实现和编写程序实现。本章内容主要介绍通过可视化界面环境完成图形用户界面的设计。

7.1.1　图形用户界面（GUI）概述

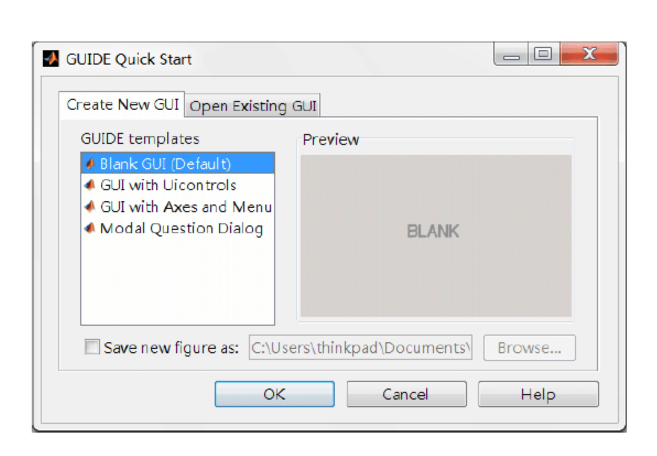
图形用户界面是指采用图形方式显示的计算机操作用户接口。与早期的计算机使用的命令行界面相比，图形界面对于用户来说在视觉上更易于接受。MATLAB 中的可视化界面环境 Guide 可以方便地创建图形界面，其功能与 Microsoft 公司的 Visual Basic、Visual C++ 类似。

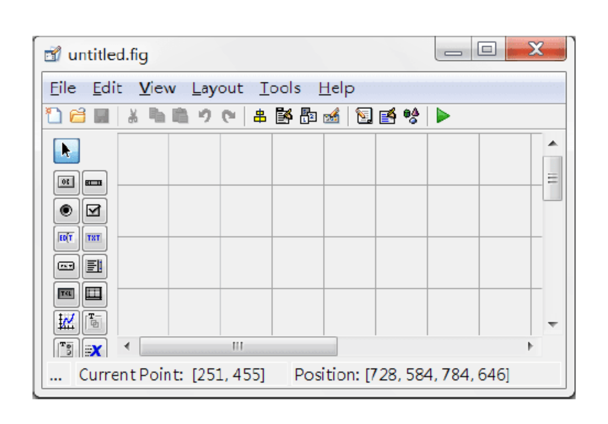
（1）用户可以通过选择菜单项打开可视化界面环境：依次选择菜单【File】/【New】/【GUI】命令，则出现“Guide Quick Start”界面，如图 7-1 所示。

（2）用户也可以在命令窗口中输入“Guide”或“Guide Filename”命令打开“Guide Quick Start”界面。

在“Guide Quick Start”界面中的“Create New GUI”选项卡可以选择“Blank GUI（Default）”创建空白的可视化图形文件，选中“Blank GUI（Default）”后单击“OK”按钮则出现空白的可视化界面，如图 7-2 所示。

也可以在“Create New GUI”选项卡中选择“GUI with Uicontrols”选项来创建具有控件的可视化界面；选择“GUI with Axes and Menu”选项来创建具有坐标轴和菜单的可视化界面；选择“Modal Question Dialog”来创建具有模态问题对话框的可视化界面。





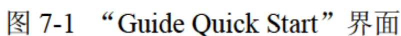

图 7-1 “Guide Quick Start”界面

图 7-2 可视化界面

7.1.2 GUIDE 的控件

1. 常用控件

控件是可视化界面的重要组成部分之一，如图 7-2 所示的图形对象面板中有按钮、切换按钮、单选按钮、复选框、文本框、静态文本框、滚动条、框架、列表框、弹出式菜单及坐标轴等各种控件，在设计界面时可根据具体要求选择所需控件。

表 7-1 中列出了一些常用控件的功能。

表 7-1 常用控件

控 件 名	Property Name	功 能 描 述
按钮	PushButton	响应用户的鼠标单击，按钮上的说明文字标识其作用
切换按钮	ToggleButton	单击时会进行凹凸状态切换
单选按钮	RadioButton	通常成组出现，单击时会以黑白点进行切换，多个单选按钮间互斥，一组中只能选中一个
复选框	CheckBox	通常成组出现，单击时会以“√”切换，具有选中、不选中和不确定等状态，可同时选中多个复选框
文本框	EditText	可随意输入、编辑单行或多行文字，并显示出来
静态文本框	StaticText	显示文字信息，不可以输入
滚动条	Slider	以图示方式显示一个范围内某数值的大概位置，数值可以通过移动滚动条进行改变
框架	Frame	规范界面，以框架将控件分组
列表框	ListBox	显示下拉文字列表，用户可以选择列表中的一项或多项
弹出式菜单	PopupMenu	类似于文本框与列表框的组合，可在下拉列表中选择
坐标轴	Axes	绘制坐标轴
面板	Panel	放置其他控件的容器
按钮组	Button Group	用于将单选按钮、复选框等分组的容器
ActiveX 控件	ActiveX Control	添加其他应用程序的 ActiveX 控件

2. 创建控件

创建控件有两种方法，第一种方法是在可视化界面环境中创建。在图形对象面板中将所需控件拖曳至空白的界面编辑面板中即可，如图 7-3 所示为各种控件在可视化环境中的

显示；第二种方法是通过 MATLAB 命令创建。通过 ui control 命令进行创建，语法如下。

```
h-control=ui control(h-Parent;PropertyName;ProperValue)
```

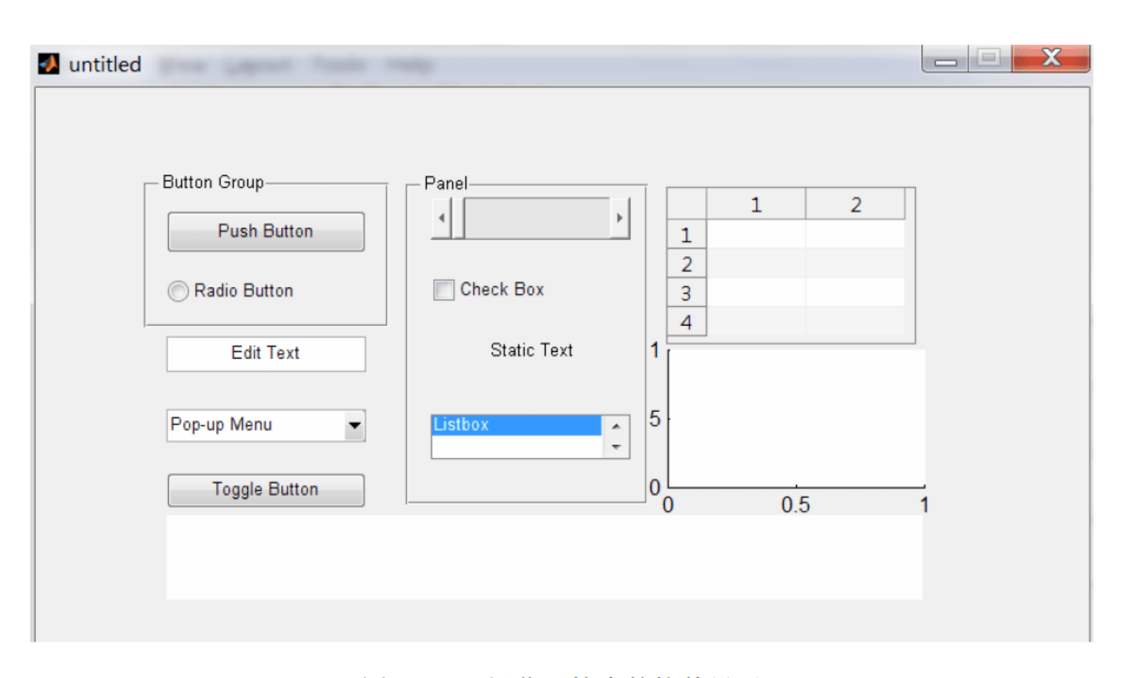


图 7-3　可视化环境中的控件显示

3. 控件属性

在创建控件后，需要设置控件的属性，由于不同控件具有的属性有所差异，这里仅介绍大部分控件都具有的属性，各控件的特有属性可查阅 MATLAB 帮助文档。

（1）string 属性：显示控件上有提示或说明作用的字符串。

（2）callback 属性：用于回调函数。

（3）enable 属性：若为“on”，则该控件有效；若为“off”，则该控件失效。

（4）tooltipstring 属性：字符串；当鼠标停留在控件上时所显示的提示信息。

（5）字体属性：包括字体名称（fontname）、字体大小（fontsize）等。

（6）interruptible 属性：指定可否中断该回调函数的执行转而执行其他函数。

7.1.3　GUIDE 开发环境

在可视化界面环境的左侧是对象控制面板，提供各种可供选择的控件。可以通过拖放控件在界面编辑面板上创建控件。在工具栏中主要提供了对象对齐工具（Align Objects）、菜单编辑器（Menu Editor）、Tab 顺序编辑器（Tab Order Editor）、工具栏编辑器（Toolbar Editor）、M 文件编辑器（M-file Editor）、属性编辑器（Property Inspector）和对象浏览器（Object Browser）等，如图 7-4 所示。

属性编辑器、对象对齐工具和对象浏览器是可视化界面环境中较常用的工具，可以在工具栏中点击按钮打开，也可以通过在菜单栏中选择相应的菜单项打开。在工具栏中点击快捷按钮或在菜单栏中通过【Tools】/【Align Objects…】命令可打开对象对齐工具，如图 7-5 所示。同样，点击快捷按钮或在菜单栏中通过【View】/【Object Browser】命令，如图

7-6 所示。单击快捷按钮或在菜单栏中通过【View】/【Property Inspector】命令，查看属性编辑器中的各属性，如图 7-7 所示。

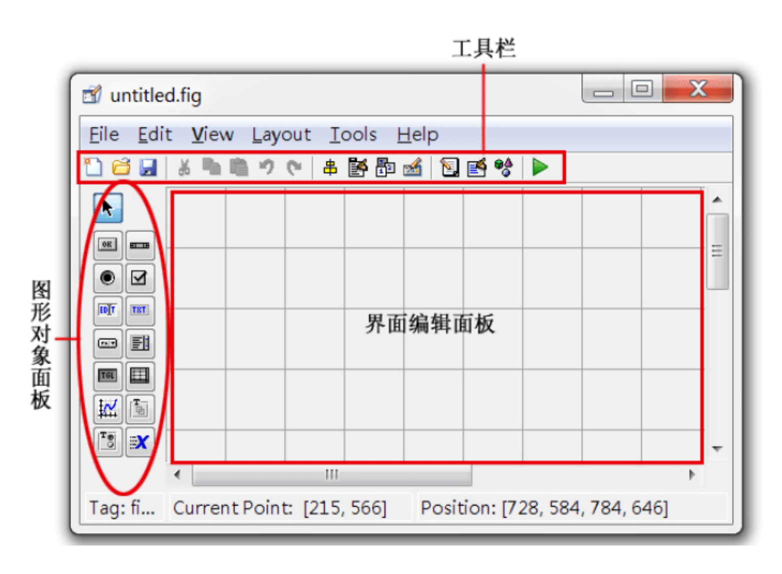


图 7-4　可视化界面

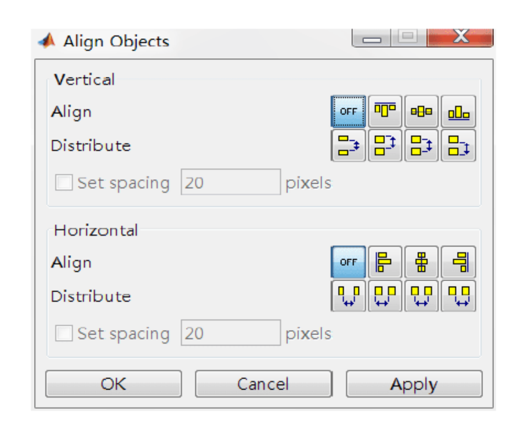


图 7-5　对象对齐工具

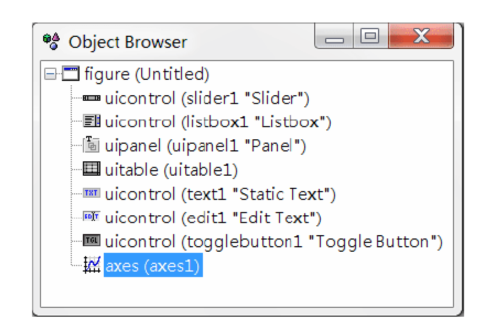


图 7-6　对象浏览器

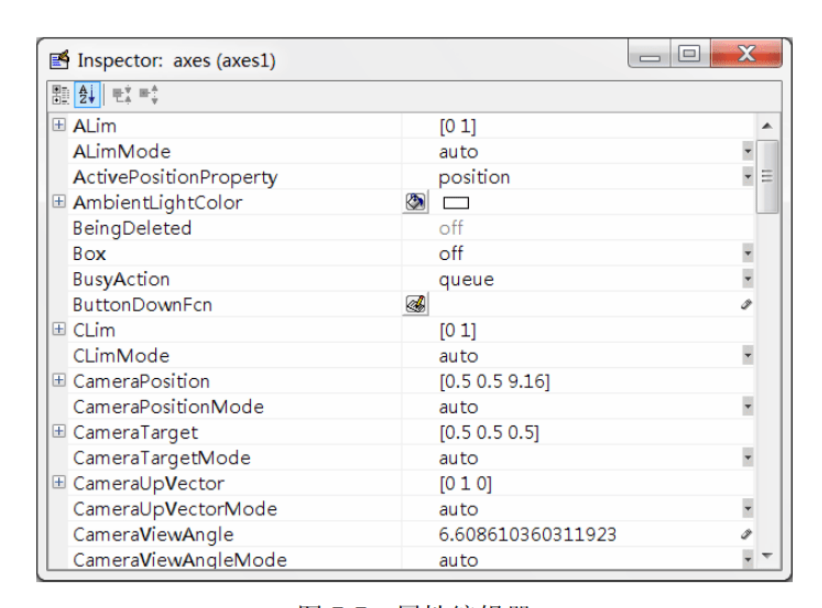


图 7-7　属性编辑器

7.2 程序设计

本节着重介绍编程设计 GUI，包括的内容有对象的回调函数、程序的一般结构、对象属性的访问、对象间数据传递、GUI 与 M 文件的数据交互及 GUI 与 Simulink 仿真的数据交互。

7.2.1 对象的回调函数

实现 GUI 的基本机制是对控件的 Callback 属性编程。如图 7-8 中所示，当创建了一个按钮控件后，单击选中该按钮后，选择【View】/【View Callbacks】命令会出现“Callback”、“CreateFcn”、“DeleteFcn”、“ButtonDownFcn” 和 “KeyPressFcn” 子菜单项，“Callback” 外的其他菜单项也是用于编写回调函数的，不同的控件可编写的回调函数会有不同。

（1）Callback：用户激活控件时执行的与控件相关的标准回调函数。

（2）CreateFcn：当创建对象时执行。

（3）DeleteFcn：当删除对象时执行。

（4）ButtonDownFcn：单击控件时执行。

（5）KeyPressFcn：当键盘按键被按下时执行。

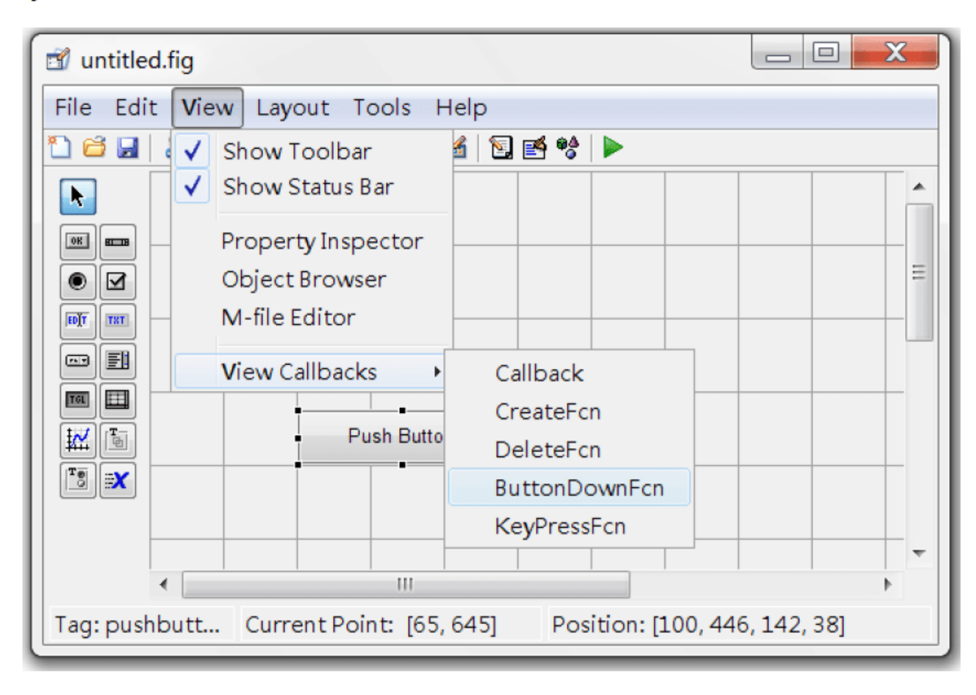


图 7-8　创建回调函数

当选择任一回调函数时，则 M 文件编辑器/调试窗口会弹出，显示出具有相应函数名的程序段，如图 7-9 所示。用户可以编写程序完善相应回调函数的功能。

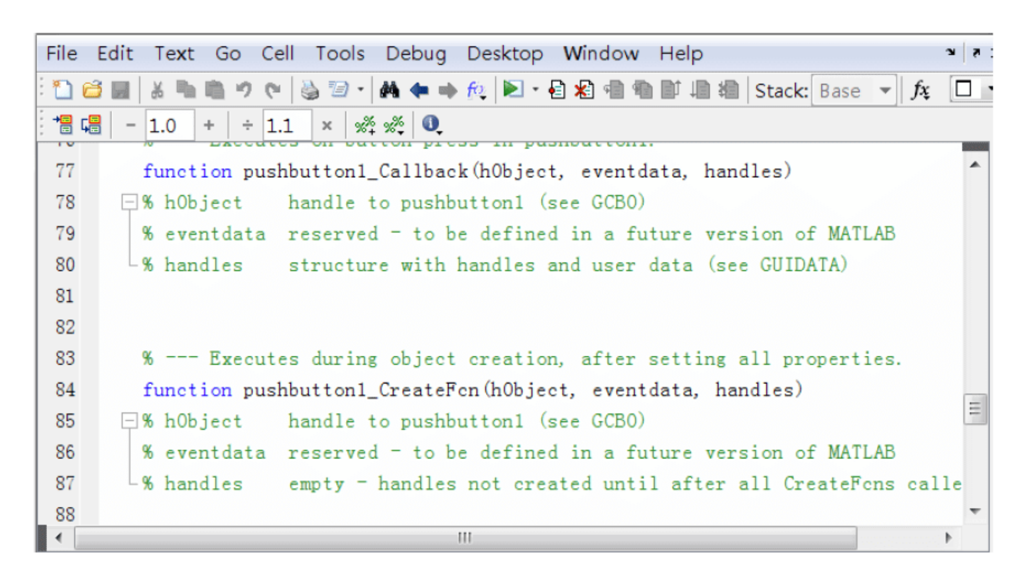


图 7-9　回调函数程序段

7.2.2　程序的一般结构

按照程序设计的观点，任何算法功能都可以通过由程序模块组成的顺序结构、选择结构和循环结构的组合来实现。

MATLAB 作为一种高级应用软件，不但可以在命令窗口中编写行命令，还可以生成自己的程序文件，要充分发挥 MATLAB 的功能，掌握 MATLAB 的程序设计是十分必要的。

7.2.3　对象属性的访问

对于对象属性的访问包括读取和设置两类，同时包括访问自身属性和其他对象属性两类。下面以 7.2.2 节中的代码为例说明访问方法。

1）访问自身属性

density_Callback 子函数中的代码 get(hObject,'string')表示读取自身的 String 属性值，并可以将返回值赋予变量，其中，str2double 函数实现字符串到数值的转换；代码 set(hObject,'String',0)表示将自身的 String 属性值设置为 0。

2）访问其他对象属性

initialize_gui 子函数中的代码 set(handles.text4,'String','lb/cu.in')表示设置图形窗口中 Tag 属性值为 text4 的对象的 String 属性值为 lb/cu.in。同理，get(handles.text4,'String')表示读取图形窗口中 Tag 属性值为 text4 对象的 String 属性值。

【例 7-1】 实现两个按钮的交互，具体步骤如下。

（1）利用空白模板创建如图 7-10 所示的界面，并保存为文件 ex0701.fig。

（2）利用属性编辑器将两个按钮的 Tag 属性值分别手工改为“PUpper”和“PDown”。

（3）设置上方按钮的 Callback 回调函数，实现先将上方按钮的 String 属性值增加“Upper”，再将下方按钮的 String 属性值增加“Down”，回调函数的内容如下所示。

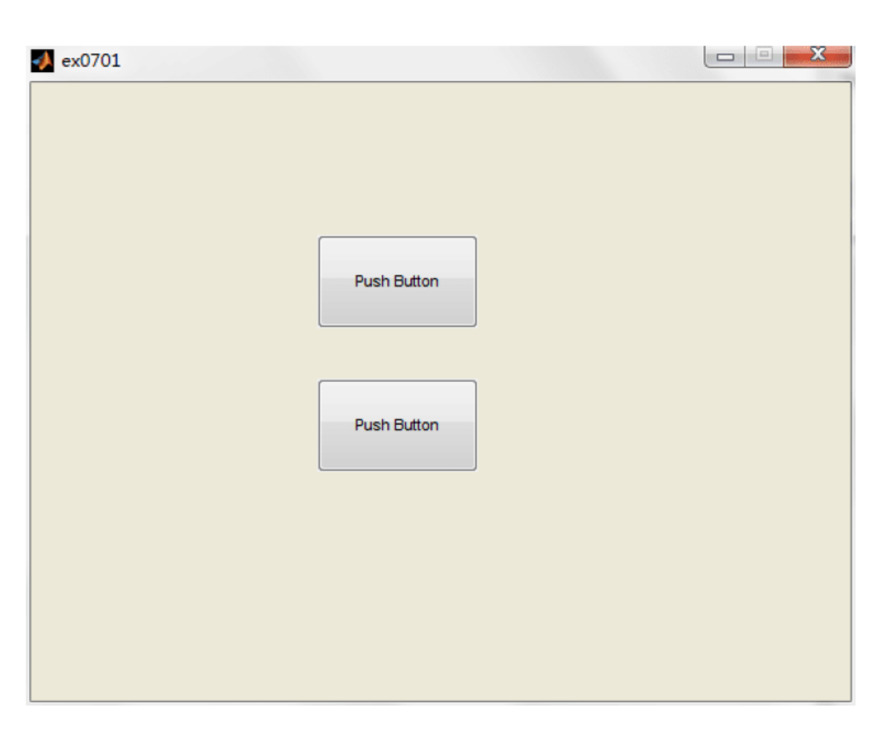


图 7-10　按钮交互界面

```
function PUpper Callback(hObject,eventdata,handles)
%hObject    handle to PUpper(see GCBO)
%eventdata  reserved-to be defined in a future version of MATLAB
%handles    structure with handles and user data(see GUIDATA)
oldUS = get(hObject,'String');
newUS = strcat(oldUS,'(Upper)');
set(hObject,'String',newUS);
oldDS = get(handles.PDown,'String');
newDS = strcat(oldDS,'(Down)');
set(handles.PDown,'String',newDS);
```

（4）激活设计的界面并单击上方按钮，可以得到如图 7-11 所示的界面。

7.2.4　对象间数据传递

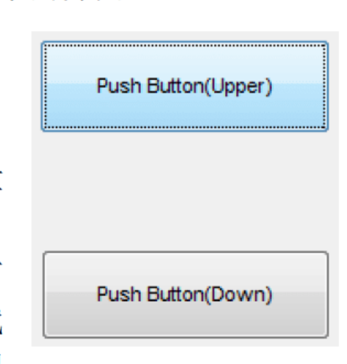


图 7-11　示例运行界面

通过前面的方法可以实现对象间属性的访问，有时还需要变量值的交互，即数据传递。

以 7.2.2 节中的代码为例，按照执行次序，变量 metricdata 首次出现在 initialize_gui 子函数的代码中，其数据类型是结构体。代码 isfield(handles,'metricdata')用于判断该变量是否存在，同时表明若存在将以 handles 字段的形式出现，即通过 handles.metricdata 调用。由于该变量为结构体，其值或属性值可以如下设置。

```
handles.metricdata.density = 0;
handles.metricdata.volume = 0;
```

density_Callback 子函数中的代码对变量 metricdata 进行数据更改，通常的做法如下：

```
handles.metricdata.volume = volume;
```

需要说明的是，在初始化或数据更改后应该更新 handles 的数据。在对象的回调函数和 OpeningFcn 与 OutputFcn 子函数中，可以使用下述语句更新。

```
guidata(hObject,handles);
```

在如 initialize_gui 的子函数中，必须将 handles 作为参数传入子函数，并且可以使用下述语句更新。

```
guidata(handles.FigureTag,handles);
```

其中，FigureTag 为对应图形窗口的 Tag 属性值，在 ex7_2 中为 figure1，即

```
guidata(handles.figure1,handles);
```

【例 7-2】 实现数据在不同控件中的传递，具体步骤如下。

（1）利用空白模板创建如图 7-12 所示的界面，并保存为 ex0702.fig。

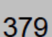


图 7-12　示例界面

（2）利用属性编辑器将两个静态文本的 Tag 属性值手工改为“STU”（上方）和“STD”（下方），String 属性值手工改为“输入 1”（上方）和“输出 2”（下方）；两个文本编辑框的 Tag 属性值手工改为“ETU”（上方）和“ETD”（下方）；两个按钮的 Tag 属性值手工改为“PBL”（左边）和“PBR”（右边），String 属性值手工改为“保存”（左边）和“计算”（右边）。

（3）设置 OpeningFcn 子函数，实现初始化，包括赋值给变量 ETUinput，将上方文本编辑框的 String 属性值设置为“ETUinput”，下方文本编辑框的 String 属性值设置为“空白”，子函数的内容如下所示。

```
function ex0902 OpeningFcn(hObject,eventdata,handles,varargin)
```

```
%Choose default command line output for ex0902
handles.output = hObject;

handles.ETUinput = 15;
set(handles.ETU,'String',handles.ETUinput);
set(handles.ETD,'String','');

%Update handles structure
guidata(hObject,handles);
```

（4）设置左边按钮的 Callback 回调函数，实现将上方文本编辑框的数据保存在变量 ETUinput 中。回调函数的内容如下所示。

```
function PB L Callback(hObject,eventdata,handles)
ETDstr = get(handles.ETU,'String');
ETDnum = str2double(ETDstr);
if isnumeric(ETDnum)
  handles.ETUinput = ETDnum;
  guidata(hObject,handles);
end
```

（5）设置右边按钮的 Callback 回调函数，实现将经过保存的变量 ETUinput 的 10 倍值写入下方文本编辑框中，回调函数的内容如下所示。

```
if isfield(handles,'ETUinput')
  set(handles.ETD,'String',10*handles.ETUinput);
end
```

（6）激活设计的界面，如图 7-13 所示。



图 7-13　激活界面 1

单击“计算”按钮，则显示如图 7-14 所示的界面。

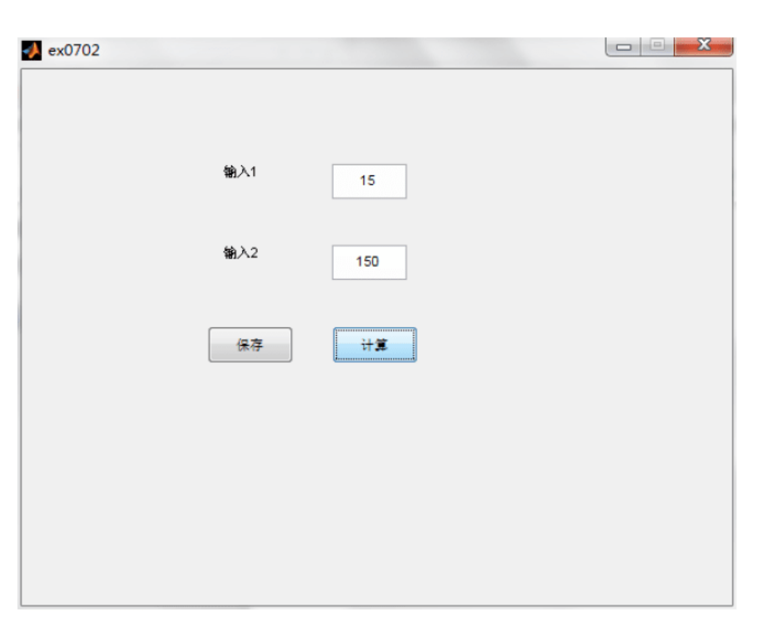


图 7-14　示例运行界面 1

在“输入 1”中重新输入 10，单击“计算”按钮，则在“输入 2”中显示的计算结果如图 7-15 所示。



图 7-15　示例运行界面 2

7.2.5　GUI 与 M 文件的数据交互

这里提到的 GUI 与 M 文件的数据交互包括如下三个含义。

（1）GUI 调用脚本或函数式的 M 文件，只需像使用 MATLAB 自带函数一样调用即可。

（2）GUI 调用工作空间中的变量，直接使用即可。

（3）工作空间调用 GUI 中的数据。

通过 evalin 函数可以实现上述功能，其具体用法如下。

```
evalin('base','expression');
 %在 MATLAB 工作空间内执行表达式 expression，其中表达式是字符串形式
vars = evalin('base',expression);
%返回在 MATLAB 工作空间内的执行结果
evalin('caller',expression);
%在调用函数工作空间内执行表达式 expression，其中表达式是字符串形式
vars = evalin('caller','expression');    %返回在调用函数工作空间内的执行结果
```

【例 7-3】 实现 GUI 与工作空间里的数据交互，具体步骤如下。

（1）利用空白模板创建界面，并保存为 ex0703.fig。

（2）利用属性编辑器将两个静态文本的 Tag 属性值手工改为“STL”（左边）和“STR”（右边），String 属性值手工改为“工作空间数据”（左边）和“图形显示结果”（右边）；列表框的 Tag 属性值手工改为“LB”；坐标轴的 Tag 属性值手工改为“AX”；两个按钮的 Tag 属性值手工改为“PBL”（左边）和“PBR”（右边），String 属性值手工改为“导入数据”（左边）和“绘制图形”（右边），设置结果如图 7-16 所示。

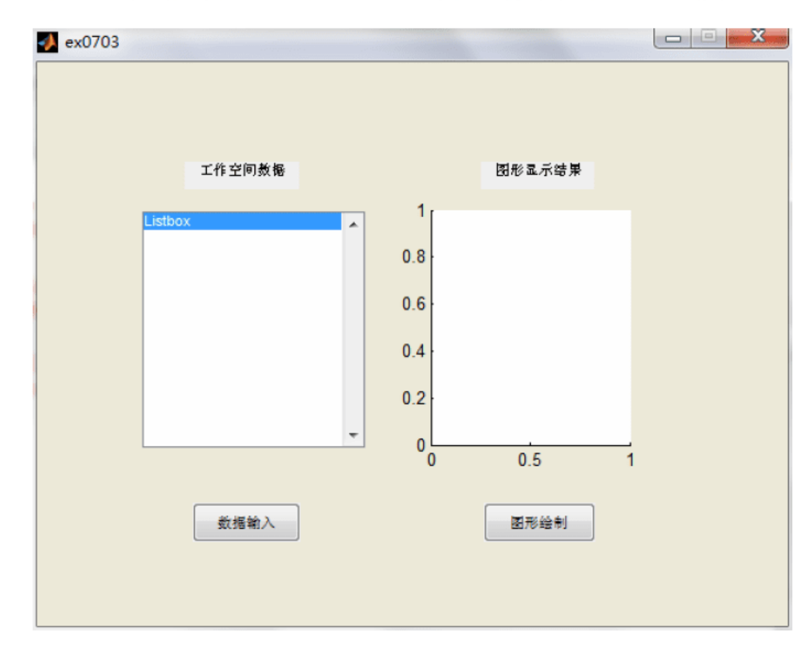


图 7-16　修改属性后的界面

（3）设置 OpeningFcn 子函数，实现初始化，包括将目前的工作空间变量导入到列表框中，并将第一个列表项作为默认列表项，子函数的内容如下所示。

```
function lb OpeningFcn(hObject,eventdata,handles,varargin)
handles.output = hObject;
update LB(handles)
guidata(hObject,handles);
```

其中，update_LB 子函数的内容如下所示。

```
function update LB(handles)
vas=evalin('base','who');
set(handles.LB,'String',vars)
set(handles.LB,'Value',1)
```

（4）设置左边按钮的 Callback 回调函数，实现将工作空间变量导入到列表框中，并将第一个列表项作为默认列表项。回调函数的内容如下所示：

```
function PBL Callback(hObject,eventdata,handles)
update_LB(handles)
```

（5）设置右边按钮的 Callback 回调函数，实现在坐标轴中绘制选中的工作空间变量，并将其值加倍后返回工作空间。回调函数的内容如下所示：

```
function PBR Callback(hObject,eventdata,handles)
axes(handles.AX)
list entries = get(handles.LB,'String');
index selected = get(handles.LB,'String');
if length(index selected) ~= 1
  errordlg('You must select one variable','Incorrect Selection','modal')
else
  var = list entries{index selected};
end
evalin('base',['plot(',var,')']);
figure
evalin('base',['plot(',var,')']);
evalin('base',[2*',var]);
```

其中，axes(handles.AX)表示选择 AX 作为绘图区域，figure 表示在 MATLAB 中生成新的图形窗口。

（6）在命令窗口输入如下语句，执行后工作空间有 5 个变量，再激活设计的界面，如图 7-17 所示。

```
clear
clc
t = 1:20;
a = tan(t);
b = cos(t);
c = exp(t);
d = sqrt(t);
```

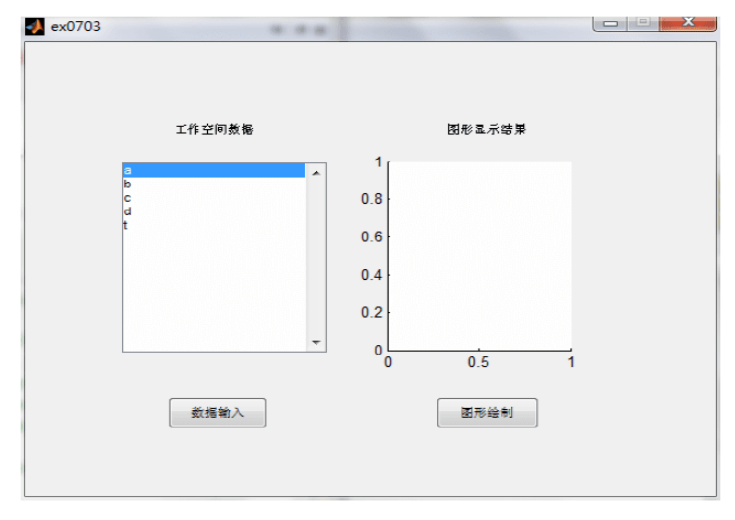


图 7-17　激活界面

选择“a”并单击右边按钮可以得到如图 7-18 所示的界面，同时在图形窗口可以得到如图 7-19 所示的图形，在命令窗口可以看到如下结果，并且工作空间增加了 ans 变量。

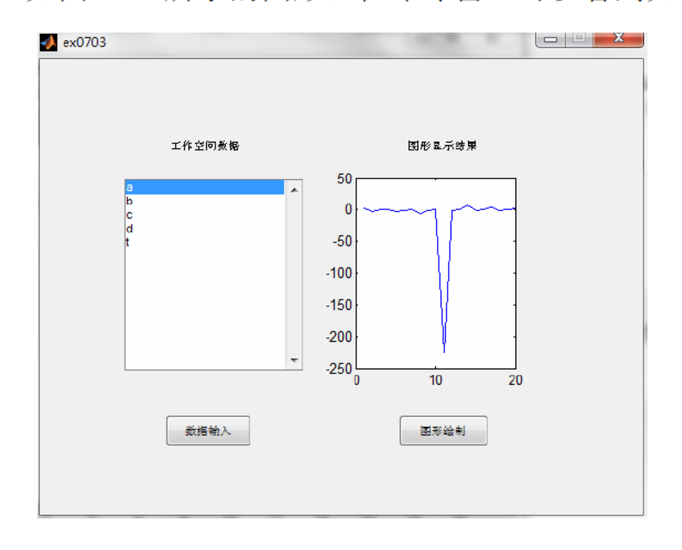


图 7-18　示例运行界面 1

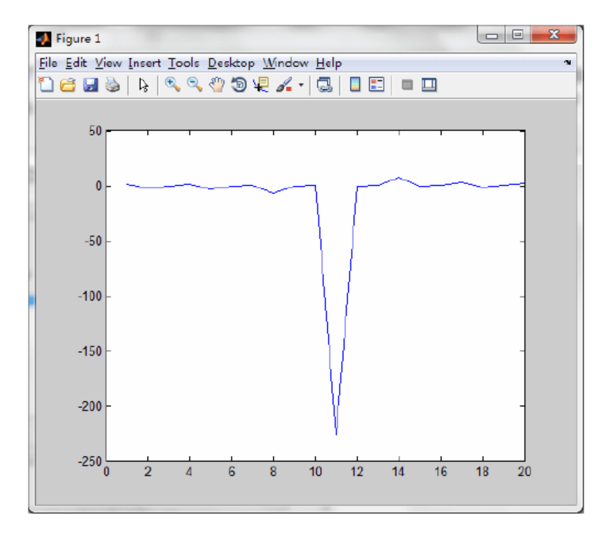
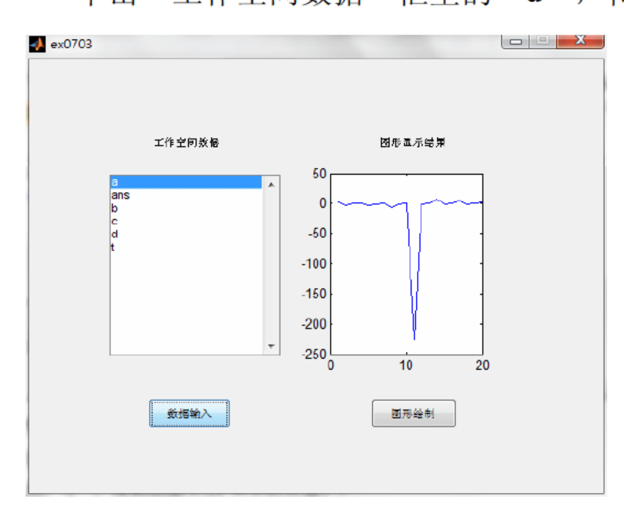


图 7-19　图形窗口运行结果

再单击左边的按钮可以得到如图 7-20 所示的界面，增加了变量 ans。

单击“工作空间数据”框里的“d”，得到如图 7-21 所示的界面。



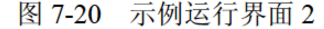


图 7-20　示例运行界面 2

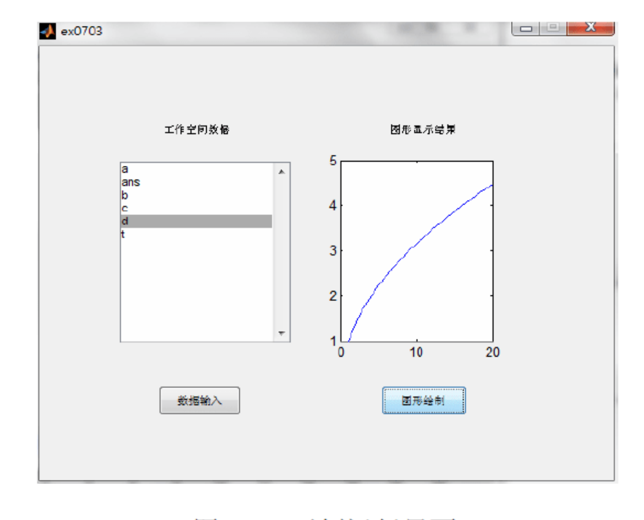


图 7-21 示例运行界面 3

7.2.6　GUI 与 Simulink 仿真的数据交互

这里提到的 GUI 与 Simulink 仿真的数据交互包括如下两个含义。

1）通过 GUI 设置 Simulink 仿真参数

设置 Simulink 仿真参数包括环境参数、模块参数、子系统中的模块参数和封装子系统参数等情况。

设置环境参数（如仿真终止时间），可以利用 MATLAB 工作空间传递参数实现，因为 Simulink 也可以利用工作空间的数据。

设置模块参数和子系统中的模块参数，可以通过向 MATLAB 工作空间传递参数实现，

也可以通过下述方法实现。

```
%打开 Simulink 文件
open system('SimFilename')
%为按相对路径指定的模块的指定参数赋值
set param('SimFilename/.../Blockname','Fieldname',Value')
%保存 Simulink 文件
save_system('SimFilename')
```

设置封装子系统参数，通过下述方法实现。

```
%打开 Simulink 文件
open system('SimFilename')
%为按相对路径指定的封装子系统的指定参数赋值
set param('SimFilename/Subsystemname','Paraname',Value);
%保存 Simulink 文件
save_system('SimFilename')
```

2）通过 GUI 调用 Simulink 仿真结果

当使用 ToWorkspace 模块时，其中的数据将传递到工作空间，GUI 直接调用即可。

【例 7-4】 实现 GUI 与 Simulink 仿真的数据交互，具体步骤如下。

（1）利用空白模板创建如图 7-22 所示的界面，并保存为 ex0704.fig。

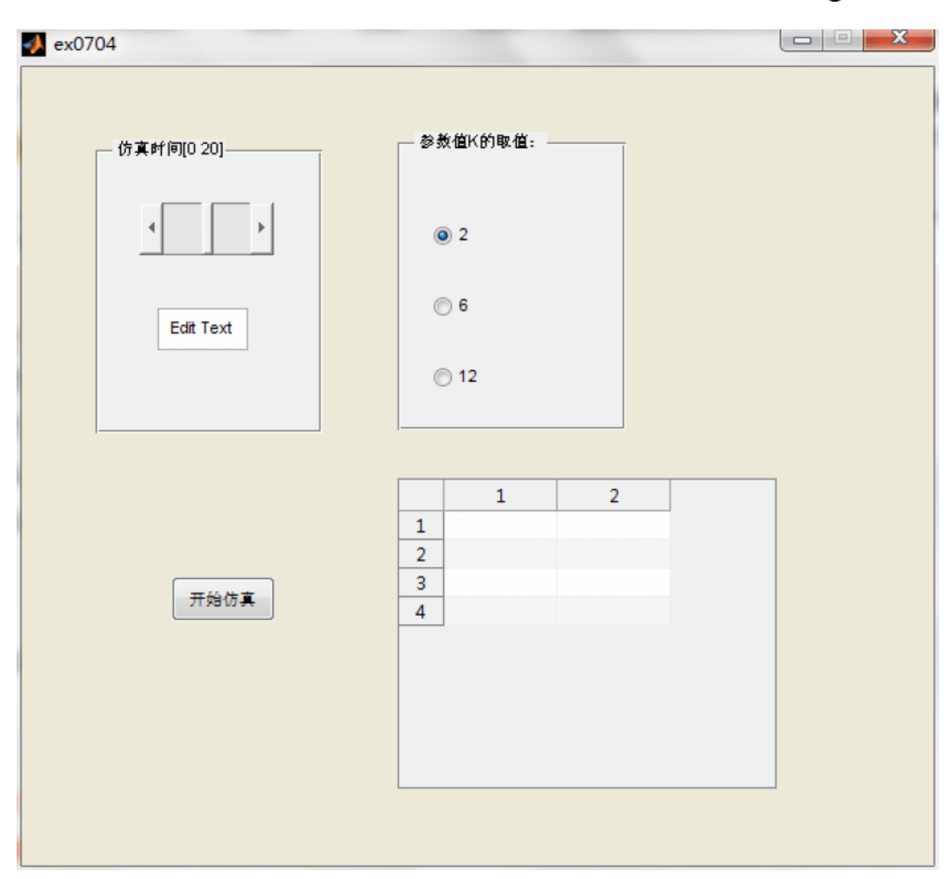


图 7-22　示例界面

（2）利用 Simulink 环境搭建如图 7-23 所示的模型，并保存为 ex07041.mdl。在命令窗

口输入 simulink 打开 simulink 库，构建所需模型。

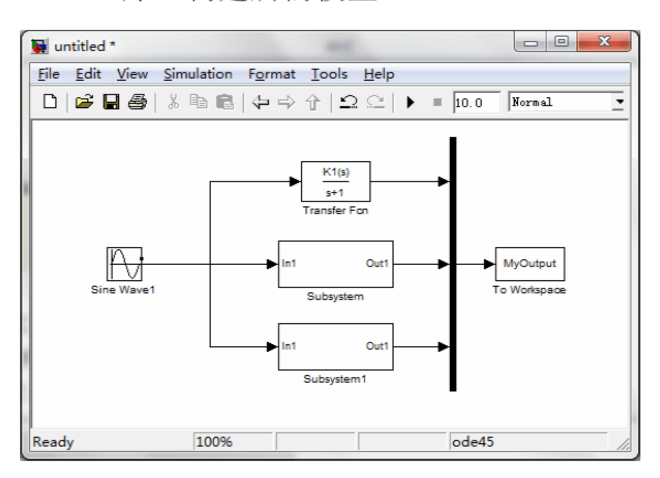


图 7-23 Simulink 模型

双击 Transfer Fcn 模块、Subsystem 子系统和 Subsystem1 封装子系统，可以分别看到如图 7-24～图 7-26 所示的界面，其中都包括增益变量（仿真取相同的值），图 7-25 和图 7-27 表示被封装的子系统，可通过右击 Subsystem1 选择“Look Under Mask”命令查看或修改封装子系统模块设置。如图 7-28 表示封装子系统的参数设置，需选中 Evaluate 属性，同时单击【Simulation】/【Configuration Parameters】菜单命令，可以看到如图 7-29 所示的参数配置界面，其中包括变量 tend，运行结果送入数组 MyOutput 中。

（3）利用属性编辑器将左上方面板的 Tag 属性值手工改为“Pan”，Title 属性值手工改为“仿真时间[0 20]”，其中滚动条的 Tag 属性值手工改为“Sli”，Max 属性值手工改为“20”，Min 属性值手工改为“0”，Value 属性值手工改为“10”，文本编辑框的 Tag 属性值手工改为“ET”；右边按钮组的 Tag 属性值手工改为“BG”，Title 属性值手工改为“参数值 K 的取值：”，其中三个单选框由上至下 Tag 属性值手工改为“RB1”、“RB2”和“RB3”，String 属性值手工改为“K=2”、“K=6”和“K=12”；按钮的 Tag 属性值手工改为“PB”，String 属性值手工改为“开始仿真”；表格的 Tag 属性值手工改为“Ta”。

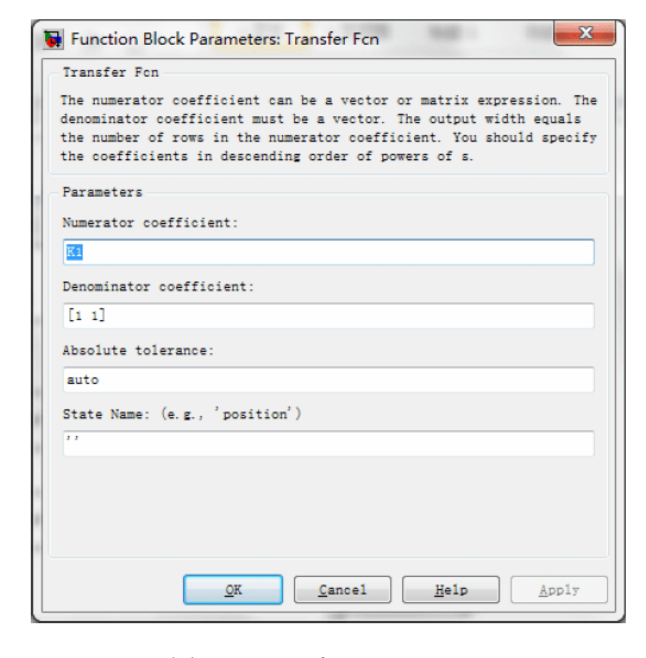


图 7-24 双击 Transfer Fcn

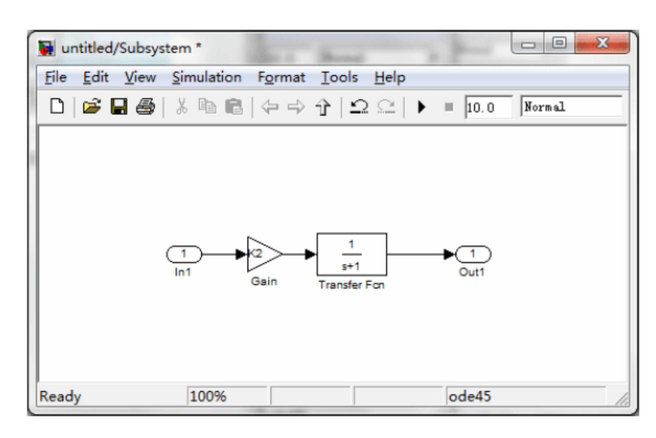
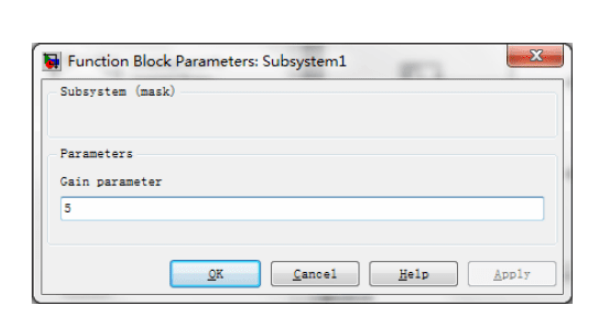
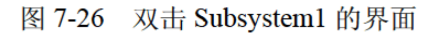


图 7-25 双击 Subsystem 的界面




图 7-26 双击 Subsystem1 的界面

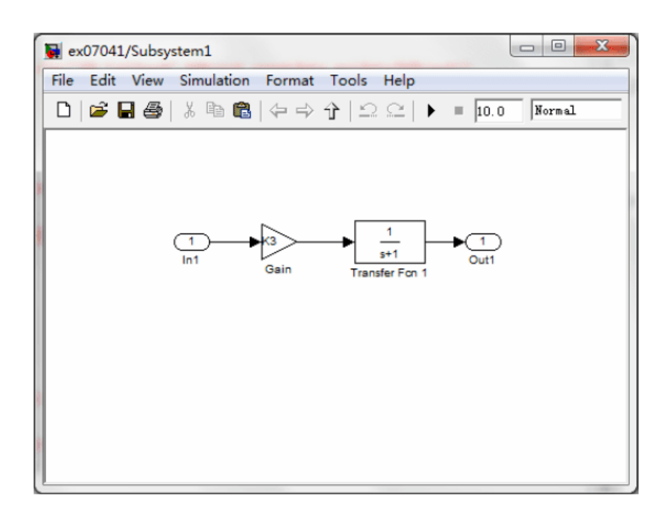


图 7-27 被封装的子系统

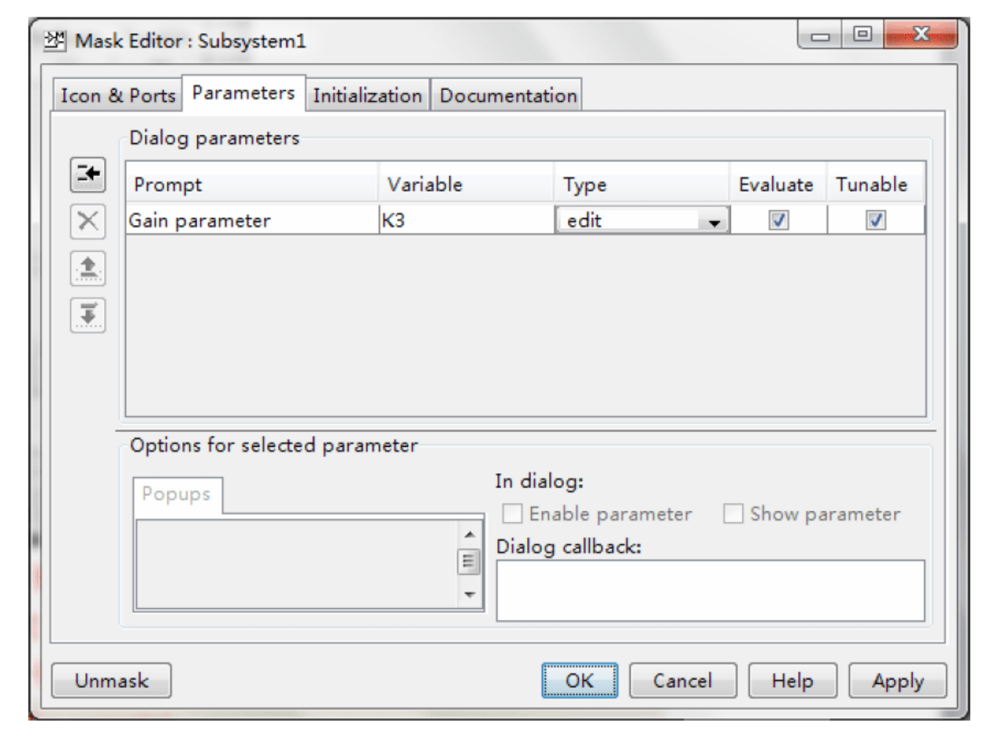


图 7-28 封装子系统的参数

（4）设置 OpeningFcn 子函数，实现初始化，包括使文本编辑框显示的数据与滚动条一致，将单选框 RB2 作为初始选择，设参数 K 的值为 6，将表格数据置空，子函数的内容如下所示。

```
function ex0704 OpeningFcn(hObject,eventdata,handles,varargin)
handles.output = hObject;
set(handles.ET,'String',get(handles.Sli,'Value'))
set(handles.BG,'SelectedObject',handles.RB2)
handles.paraK = 6;
set(handles.Ta,'Data',[])
guidata(hObject,handles);
```

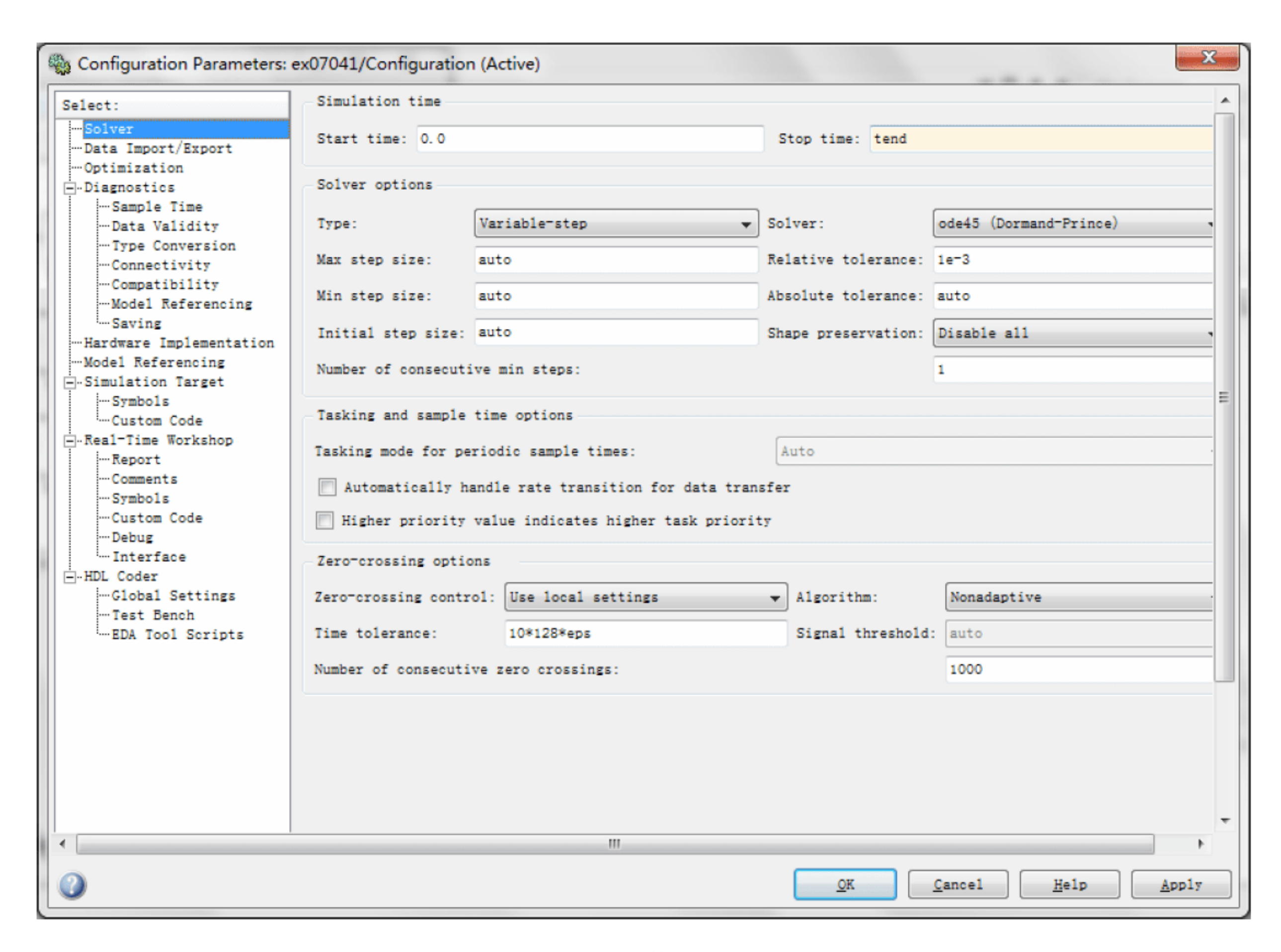


图 7-29　参数配置界面

（5）设置滚动条的 Callback 回调函数，实现将其值写入文本编辑框，回调函数的内容如下所示。

```
function Sli Callback(hObject,eventdata,handles)
set(handles.ET,'String',get(handles.Sli,'Value'))
```

（6）设置按钮的 Callback 回调函数，实现读入仿真时间和参数 K，输出参数并将其赋给 Simulink 模型，运行后将数据的最后 10 行显示在表格中，回调函数的内容如下所示。

```
function PB Callback(hObject,eventdata,handles)
tendvalue = get(handles.Sli,'Value');
evalin('base',['tend=',num2str(tendvalue)]);

switch get(handles.BG,'SelectedObject')
  case handles.RB1
    handles.paraK = 2;
  case handles.RB2
    handles.paraK = 6;
  case handles.RB3
    handles.paraK = 12;
end
K = handles.paraK;
evalin('base',['K1=',num2str(K)]);
evalin('base',['K2=',num2str(K)]);
```

```
K3 = K
open_system('ex07041')
set_param('ex07041/Subsystem1','K3',num2str(K3));
save_system('ex07041')
close_system('ex07041')
sim('ex07041');
set(handles.Ta,'Data',MyOutput(end-9:end,:))
```

（7）激活设计的界面，如图 7-30 所示。拖动滚动条并选择参数值后，单击“开始仿真”按钮可以得到如图 7-31 所示的界面。

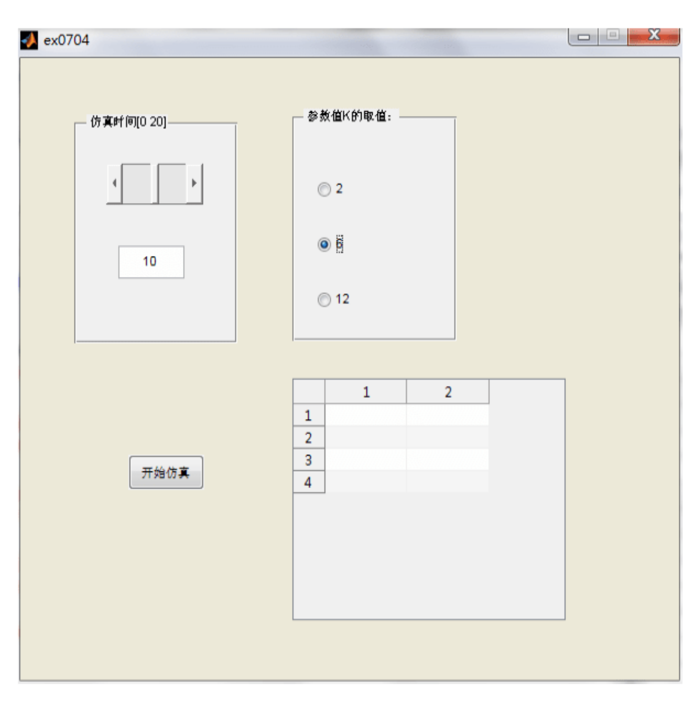


图 7-30　激活界面

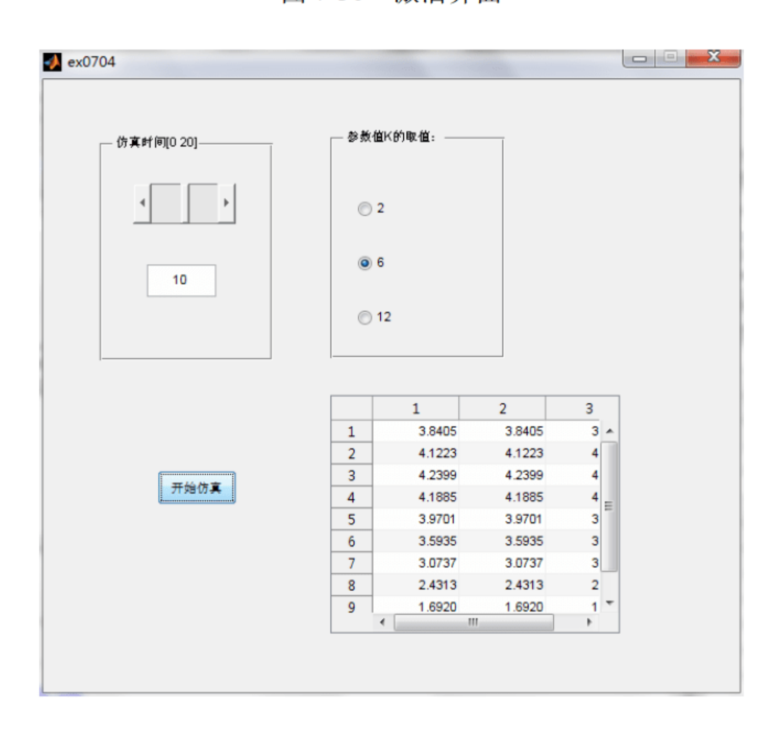


图 7-31　示例运行界面 1

（8）在文本编辑框中修改数值为 10，并选择 K=12，单击“开始仿真”按钮可以得到如图 7-32 所示的界面。



图 7-32　示例运行界面 2

7.2.7　中断执行

默认情况下，MATLAB 允许正在执行的回调函数被之后调用的回调函数中断。例如，我们建立一个对话框，用来作为加载数据的进度指示器。对话框中有一个“Cancel”命令按钮，它能够停止加载操作。“Cancel”命令按钮的响应程序将中断当前正在运行的响应程序。

所有对象都有控制其回调函数能否被中断的属性 Interruptible，默认值为 on，表示回调函数可以中断。MATLAB 中有在遇到命令 drawnow、figure、getfreame、pause 和 waitfor 时才会执行中断，转而查询事件序列，否则将会执行完回调函数。需要说明的是，图形窗口的重绘、CloseRequestFcn 和 ResizeFcn 事件，对象的 DeleteFcn 和 CreatFcn 事件可以任意中断回调函数。

当回调函数被中断时，MATLAB 先将该回调函数挂起，然后处理事件序列中的事件。同时所有对象都具有一个 BusyAction 属性，该属性决定了不允许中断的回调函数的处理方式：一是将新事件加入事件序列，等待当前回调函数执行完毕再处理；二是直接舍弃新事件。

【例 7-5】 实现回调函数的中断，具体步骤如下。

（1）利用空白模板创建如图 7-33 所示的界面，并保存为 ex0705.fig。



图 7-33　示例界面

（2）利用属性编辑器将两个静态文本的 Tag 属性值手工改为“STU”（上方）和“STD”（下方），String 属性值手工改为“按钮事件状态”（上方）和“鼠标事件状态”（下方）；按钮的 Tag 属性值手工改为“PB”，String 属性值手工改为“测试”。

（3）设置 OpeningFcn 子函数，实现初始化，包括保存初始值及设置标志，子函数的内容如下所示。

```
function ex0705_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to ex0705 (see VARARGIN)
% Choose default command line output for ex0705
handles.output = hObject;
handles.strU=get(handles.STU,'String');
handles.strD=get(handles.STD,'String');
handles.strflag=0;

% Update handles structure
guidata(hObject, handles);
```

（4）设置按钮的 Callback 回调函数，实现中断的条件，并且显示中断的过程，回调函数的内容如下所示。

```
function PB_Callback(hObject, eventdata, handles)
% hObject    handle to PB (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
 % Hint: get(hObject,'Value') returns toggle state of PB
handles.strflag=handles.strflag+1;
guidata(hObject, handles);
if handles.strflag ==1
     newstrU=[handles.strU,'启动'];
```



```
        set(handles.STU,'String',newstrU);
         newstrD=[handles.strD,'未启动'];
        set(handles.STD,'String',newstrD);
        pause
        newstrU=[handles.strU,'结束'];
        set(handles.STU,'String',newstrU);
        newstrD=[handles.strD,'结束'];
        set(handles.STD,'String',newstrD);
end
```

（5）设置图形窗口的 WindowButtonMotionFcn 回调函数，实现中断处理，回调函数的内容如下所示。

```
function figure1_WindowButtonMotionFcn(hObject, eventdata, handles)
% hObject    handle to figure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
if handles.strflag==1
        newstrU=[handles.strU,'中断'];
        set(handles.STU,'String',newstrU);
         newstrD=[handles.strD,'启动'];
        set(handles.STD,'String',newstrD);
end
```

（6）激活设计的界面，如图 7-34 所示。



图 7-34　激活界面

使用鼠标在图形窗口上滑动时界面无变化，单击“启动测试”按钮可以得到如图 7-35 所示的界面，按下键盘上的任意键可以得到如图 7-36 所示的界面。



图 7-35　示例运行界面 1



图 7-36　示例运行界面 2

7.2.8 多界面实例

前面的例子都是对于单界面的操作，下面给出一个多界面的例子。在一个界面中需要调用其他界面时，可以在某个回调函数中采用如下语句。

```
figname %界面对应的图形文件名
```

需要关闭一个界面时，只需在某个回调函数中采用如下语句。

```
delete(handles.figurename)  %figurename 为图形窗口的 Tag 属性值
```

【例 7-6】 实现多界面的操作，具体步骤如下。

（1）利用空白模板创建如图 7-37 所示的界面，并保存为 ex0706.fig。单击【Tools】/【GUI Options】菜单命令，设置参数如图 7-38 所示，即窗口大小可调整，允许同时运行多个实例（不设置此项，将无法出现 ex0706 实例）。

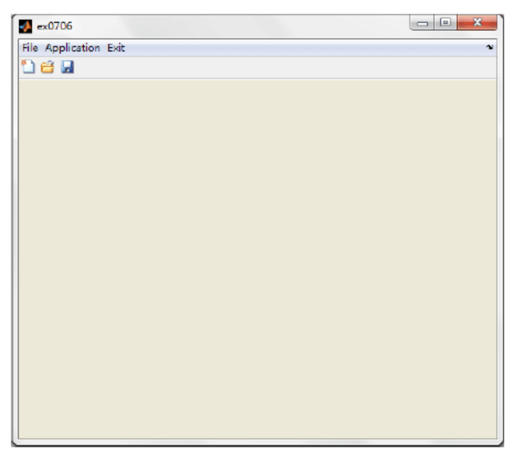


图 7-37 示例界面

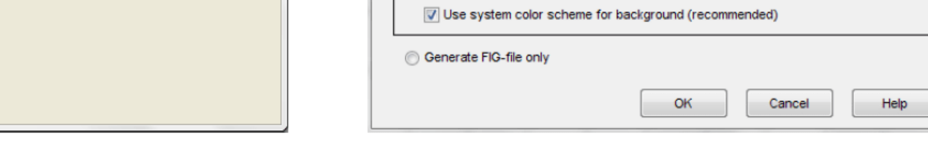


图 7-38 GUI 参数设置

（2）利用菜单编辑器创建如图 7-39 所示的菜单，利用工具条编辑器创建如图 7-40 所示的工具条，利用属性查看器得到图形窗口的 Tag 属性值为“figure1”。

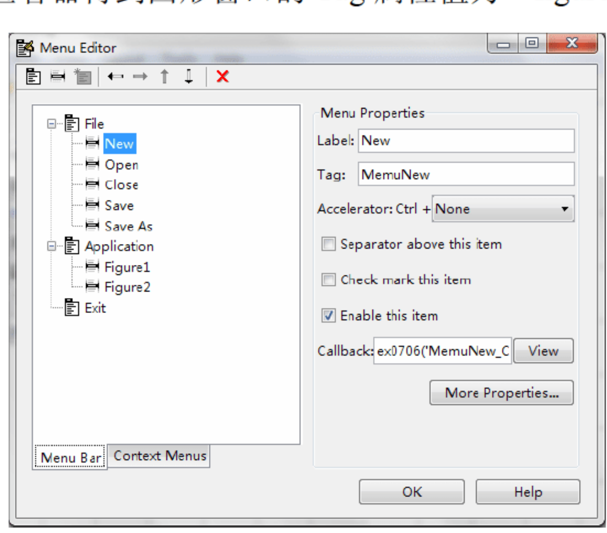


图 7-39 菜单设置



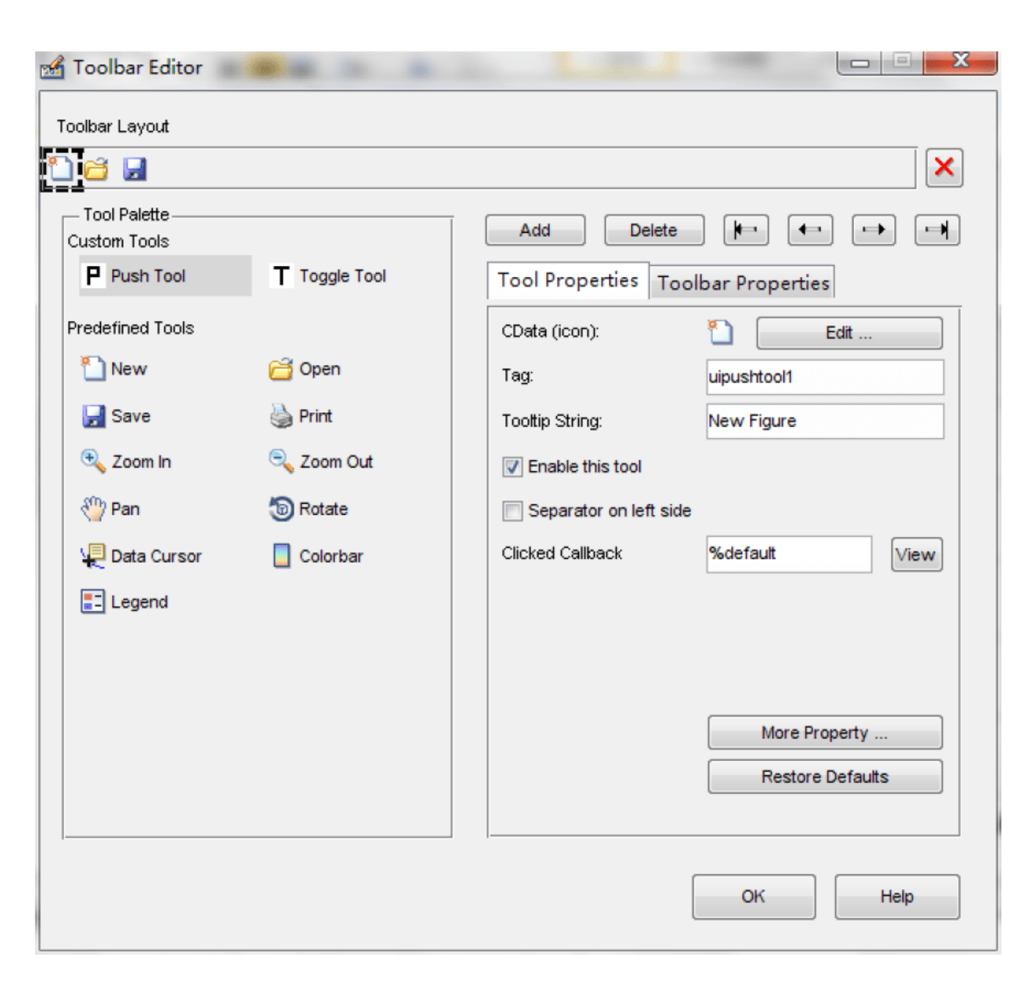


图 7-40　工具条设置

（3）利用空白模板创建如图 7-41 和图 7-42 所示的界面，并分别保存为 ex07061.fig 和 ex07062.fig。

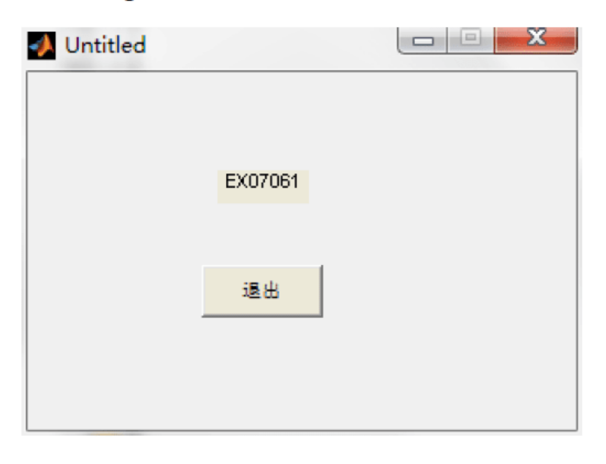


图 7-41　示例界面



图 7-42　示例界面

（4）对于 ex07061.fig，利用属性查看器得到图形窗口的 Tag 属性值为“figure1”，设置静态文本的 String 属性值为“EX07061”，按钮的 String 属性值为“退出”。对于 ex07062.fig，利用属性查看器得到图形窗口的 Tag 属性值为“figure1”，设置静态文本的 String 属性值为“EX07062”，按钮的 String 属性值为“调用并退出”。

（5）对于 ex0706.fig，设置菜单命令【Application】/【Figure1】的 Callback 回调函数，实现对 ex07061.fig 的调用和对自身的再次调用，回调函数的内容如下所示。

```
function MenuFigure1 Callback(hObject, eventdata, handles)
% hObject    handle to MenuFigure1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ex07061
ex0706
```

设置菜单命令【Application】/【Figure2】的 Callback 回调函数，实现对 ex07062.fig 的调用，回调函数的内容如下所示。

```
function MenuFigure2 Callback(hObject, eventdata, handles)
% hObject    handle to MenuFigure2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ex07062
```

设置菜单命令“Exit”的 Callback 回调函数，实现关闭 ex0706.fig，回调函数的内容如下所示。

```
function MenuExit_Callback(hObject, eventdata, handles)
% hObject    handle to MenuExit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.figure1)
```

（6）对于 ex07061.fig，设置按钮的 Callback 回调函数，实现对 ex07062.fig 的调用和自身的退出，回调函数的内容如下所示。

```
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
ex07062
delete(handles.figure1)
```

（7）对于 ex07062.fig，设置按钮的 Callback 回调函数，实现自身的退出，回调函数的内容如下所示。

```
function pushbutton1 Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
delete(handles.figure1)
```

（8）激活 ex0706.fig 界面，如图 7-43 所示，可以改变图形窗口大小。选择菜单

【Application】/【Figure2】命令后移动图形窗口，可以得到如图 7-44 所示的界面。

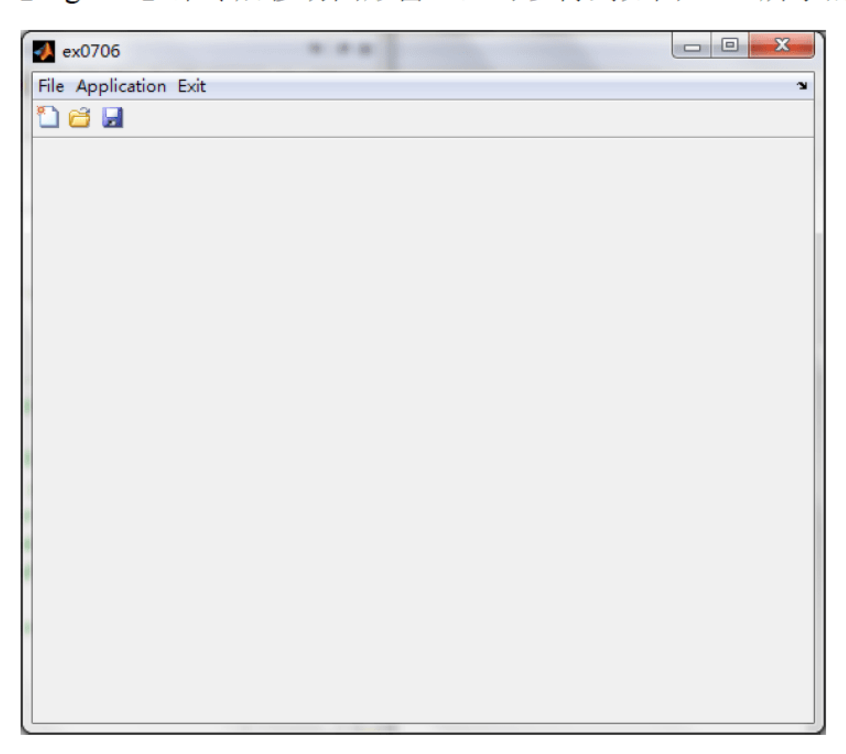


图 7-43　激活界面

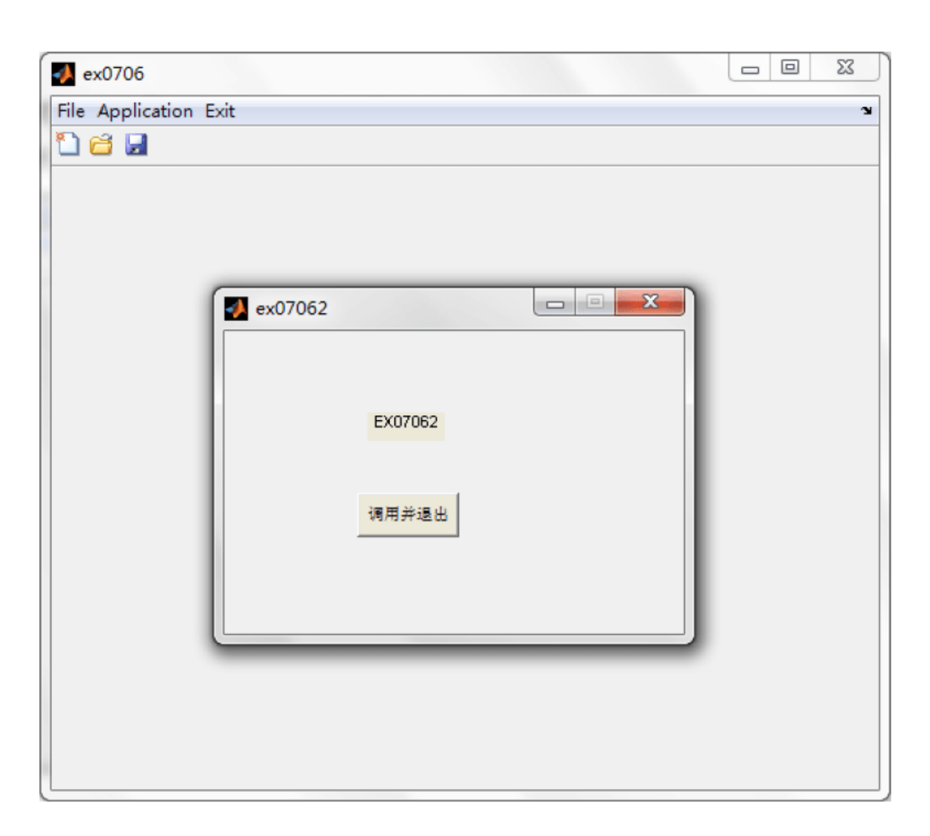


图 7-44　示例运行界面 1

激活 ex07061.fig 界面，如图 7-45 所示。

单击 ex07061 中的“退出”按钮，可以得到如图 7-46 所示的界面。

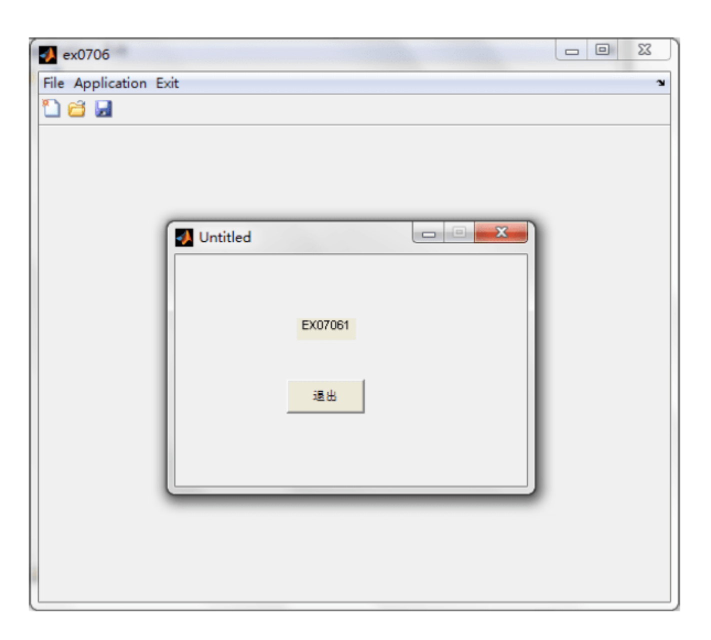


图 7-45　激活界面



图 7-46　示例运行界面 2

选择 ex0706 中的菜单命令“Exit”，可以得到如图 7-47 所示的界面，再选择 ex0906 中的菜单命令“Exit”，屏幕上将没有图形窗口。

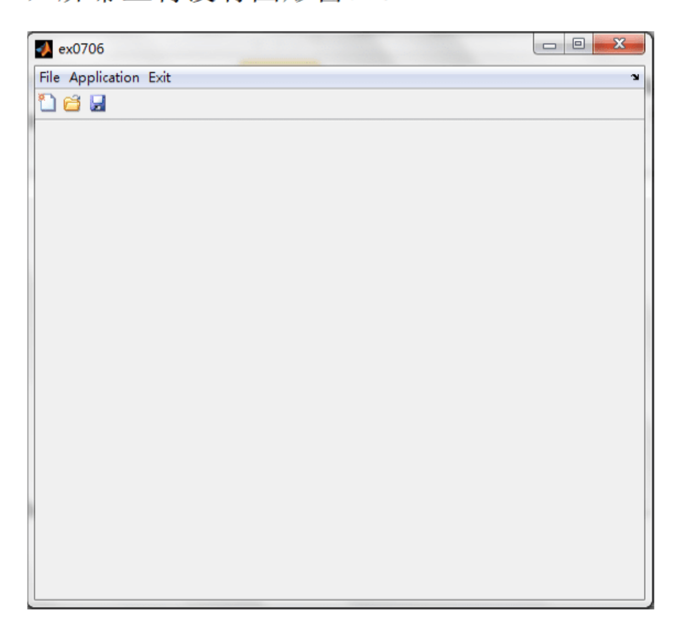


图 7-47　示例运行界面 3

7.3　GUI 应用

通过前面两节的介绍，特别是 7.2 节中的例子，使我们了解到使用 GUIDE 可以设计出界面精美和功能强大的 GUI。尽管与其他编程软件相比还有一点差距，如控件组中没有常见的下拉框、组合框等，但可以通过现有控件的组合和代码的编写实现相应功能。MATLAB 主要用于科学计算和模型仿真，现有的 GUI 设计能力应该能够满足要求。

下面首先介绍 GUI 设计的一般步骤，其次通过一个例子给出 GUI 对应的完整 M 文件。

7.3.1 GUI 设计的一般步骤

前面 7.2 节中的各个例子也体现了 GUI 设计的一般步骤，主要内容如下。

（1）利用指定模板创建初始界面，在界面上布局控件、菜单和工具条，可以充分利用 MATLAB 提供的界面设计器、菜单编辑器与工具条编辑器设计出精美的界面。

（2）利用属性编辑器、菜单编辑器及工具条编辑器为每个对象赋予属性值，最重要的属性是 Tag，它将作为该对象的标识出现在对象浏览器和 M 文件编辑器中。

（3）利用 M 文件编辑器编写初始化函数、结束自函数、对象回调函数及使用到的子函数，设计出具有强大功能的 GUI。

（4）利用 M 文件的调试方法得到正常运行的 GUI。

7.3.2 GUI 设计实例

本节用一个简单的实例，详细讲解 GUI 的设计过程。

【例 7-7】 初级 GUI 编程实例，使用 GUIDE 进行界面设计，绘制分析信号的频率和周期的图形。

- ❑ 建立 6 个静态文本，用于显示函数和标注相应控件的提示。
- ❑ 建立 2 个坐标轴对象，用于显示周期和事件的图形。
- ❑ 建立 3 个文本编辑框，用于输入数据。
- ❑ 建立一个按钮用于绘制图形。

针对上述步骤，完成的界面如图 7-48 所示，保存为 singal.fig 文件。

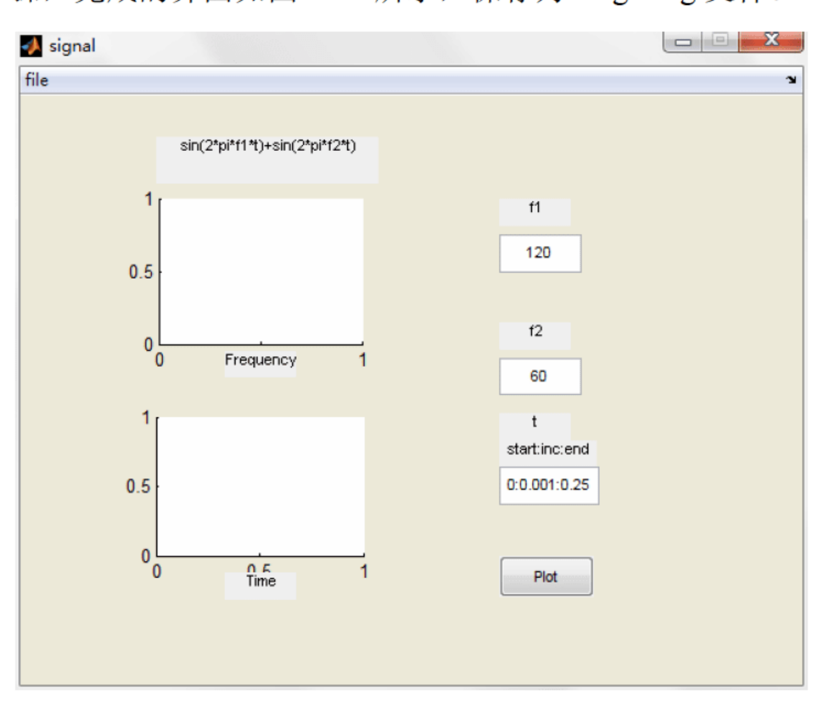


图 7-48 示例界面 1

设置控件的相关属性，每个控件在创建时都会由开发环境自动产生一个标识（Tag），在程序设计中，为了编辑、记忆和维护的方便，一般为控件设置一个新的标识。

设置一个坐标轴的标识为 frequency_axes，用于显示频率图形，如图 7-49 所示。

- ❑ 设置第二个坐标轴的标识为 time_axes，用于显示时域图形。
- ❑ 三个文本编辑框的标识为 f1_input、f2_input 和 t_input，分别用于输入两个频率和自变量时间的间隔，如图 7-50 所示。

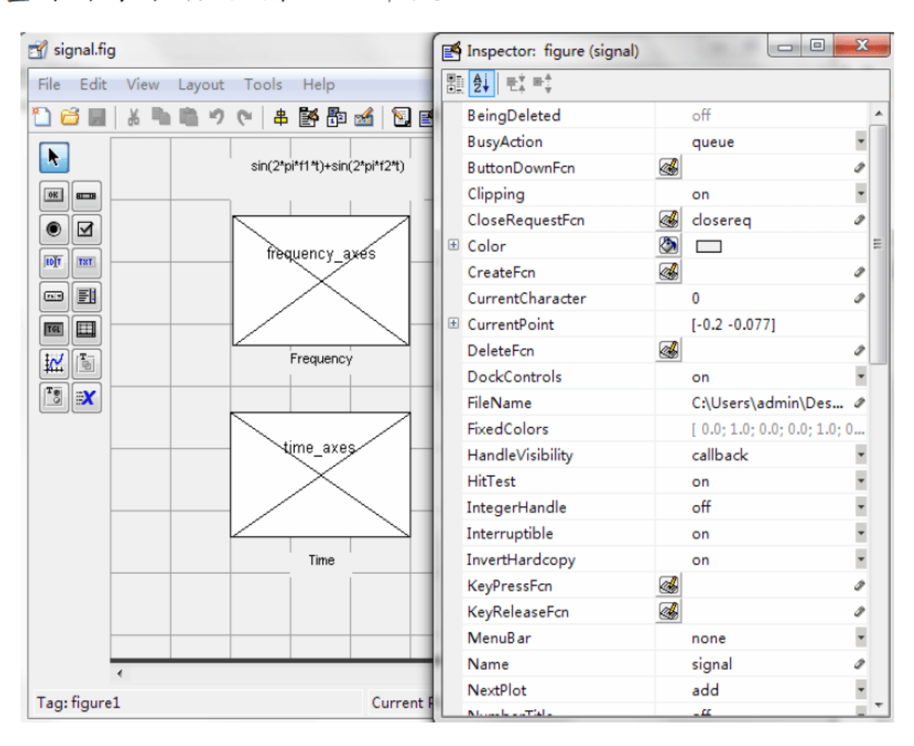
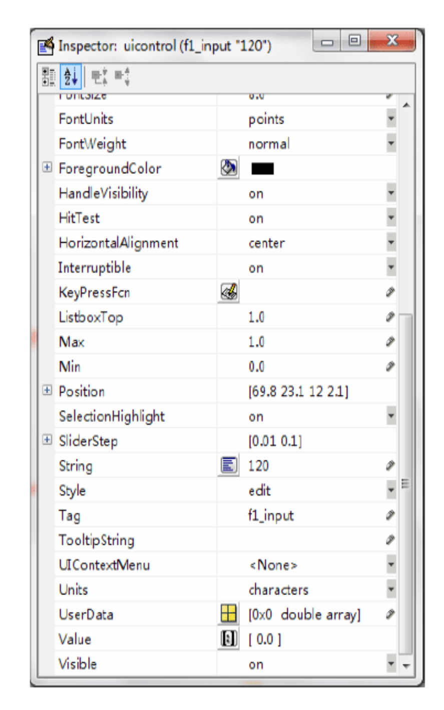
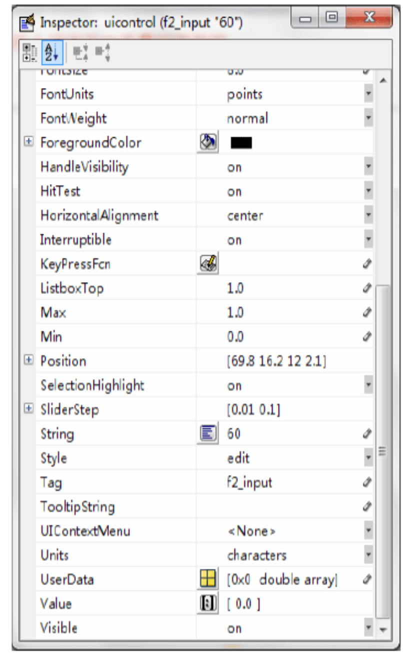


图 7-49　坐标轴的设置





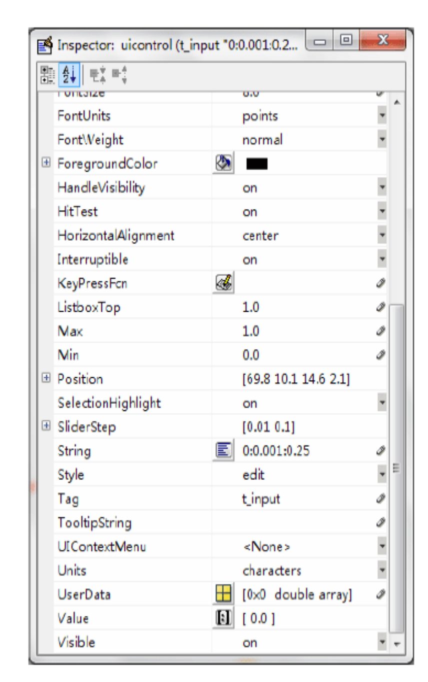


图 7-50　文本编辑框的设置

另外，设置其他几个静态文本控件和按钮的名称，单击【Property Inspector】/【String】菜单命令，设置显示名称，如图 7-51 所示。

添加菜单，如图 7-52 所示。

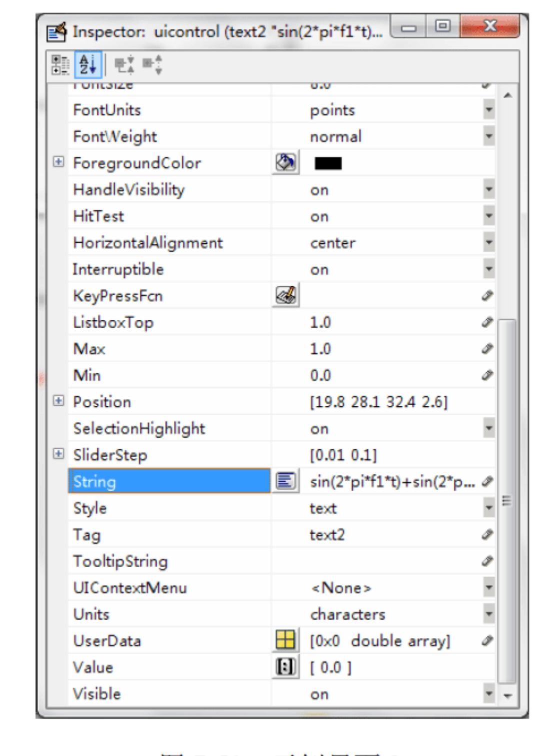


图 7-51　示例界面 2

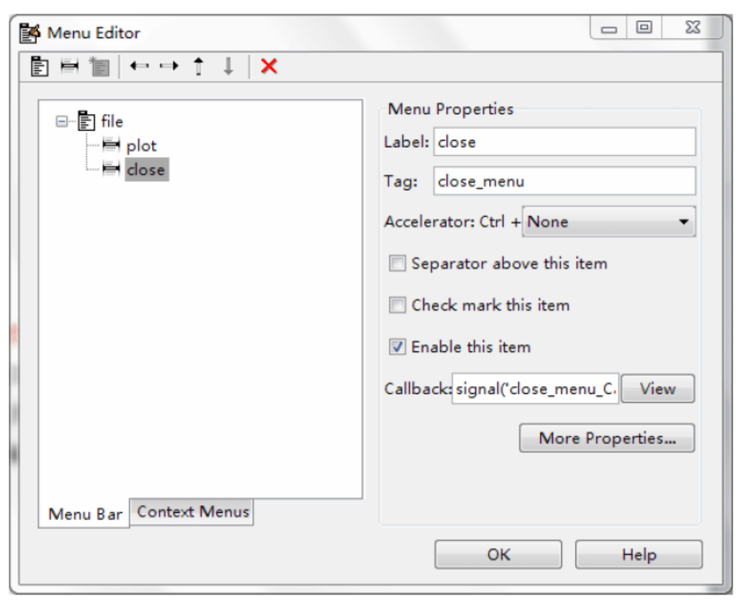


图 7-52　添加菜单

如图 7-52 所示，建立一级菜单 file，设置两个子菜单项 plot 和 close。

- ❑ 子菜单项 plot 的 Tag 设置为“plot_menu”，调用绘图功能。
- ❑ 子菜单项 close 的 Tag 设置为“close_menu”，执行关闭图形的功能。

编写代码完成程序中变量的赋值、输入、输出及绘图等工作，打开 signal.m 文件，生成的代码如下。

```
function varargout = signal(varargin)
% SIGNAL M-file menu for signal.fig
gui Singleton = 1;
gui_State = struct('gui_Name',      mfilename, ...              %GUI 结构
                'gui Singleton',  gui Singleton, ...
                'gui OpeningFcn', @signal OpeningFcn, ...
                'gui OutputFcn',  @signal OutputFcn, ...
                'gui LayoutFcn',  [] , ...
                'gui Callback',   []);
if nargin && ischar(varargin{1})
     %输入参数判断处理
      gui State.gui Callback = str2func(varargin{1});
end

if nargout
 %输出参数判断处理
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
```

```
else
     gui mainfcn(gui State, varargin{:});
end
% End initialization code

% --- Executes just before signal is made visible.
function signal OpeningFcn(hObject, eventdata, handles, varargin)
% Choose default command line output for signal
handles.output = hObject;
   % Update handles structure
guidata(hObject, handles);

% --- Outputs from this function are returned to the command line.
function varargout = signal OutputFcn(hObject, eventdata, handles)
% Get default command line output from handles structure
varargout{1} = handles.output;

function f1 input Callback(hObject, eventdata, handles)
% Hints: get(hObject,'String') returns contents of f1 input as text
% str2double(get(hObject,'String')) returns contents of f1 input as a double
% --- Executes during object creation, after setting all properties.
function f1 input CreateFcn(hObject, eventdata, handles)
% Hint: edit controls usually have a white background on Windows.
%See ISPC and COMPUTER.
set(gcbo,'String','120')
if ispc&&isequal
(get(hObject,'BackgroundColor'),get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
end

function f2 input Callback(hObject, eventdata, handles)
% Hints: get(hObject,'String') returns contents of f2 input as text
% str2double(get(hObject,'String')) returns contents of f2 input as a double
% --- Executes during object creation, after setting all properties.
function f2 input CreateFcn(hObject, eventdata, handles)
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
set(gcbo,'String','60')
if ispc && isequal
(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
      set(hObject,'BackgroundColor','white');
end

function t_input_Callback(hObject, eventdata, handles)
% Hints: get(hObject,'String') returns contents of t_input as text
% str2double(get(hObject,'String')) returns contents of t_input as a double

% --- Executes during object creation, after setting all properties.
function t_input_CreateFcn(hObject, eventdata, handles)
% Hint: edit controls usually have a white background on Windows.
% See ISPC and COMPUTER.
```

```
set(gcbo,'String','0:0.001:0.25')
if ispc && isequal
(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
       set(hObject,'BackgroundColor','white');
end

% --- Executes on button press in plot pushbutton.
function plot pushbutton Callback(hObject, eventdata, handles)
% --------------------------------------------------------------------
function file menu Callback(hObject, eventdata, handles)
% --------------------------------------------------------------------
function plot menu Callback(hObject, eventdata, handles)
% --------------------------------------------------------------------
function close menu Callback(hObject, eventdata, handles)
close
```

调用 plot_pushbutton_Callback 执行绘制图形的功能，函数代码如下：

```
% hObject    handle to plot pushbutton (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
f1=str2double(get(handles.f1 input,'String'));
f2=str2double(get(handles.f2_input,'String'));
t=eval(get(handles.t_input,'String'));
x=sin(2*pi*f1*t)+sin(2*pi*f2*t);     %计算数据
y=fft(x,512)
m=y.*conj(y)/512;
f=1000*(0:256)/512;
axes(handles.frequency_axes)         %选择显示周期的坐标轴
plot(f,m(1:257))
set(handles.frequency axes,'XMinorTick','on')
grid on
axes(handles.time_axes)              %选择显示时间的坐标轴
plot(t,x)
set(handles.time axes,'XMinorTick','on')
grid on
调用 f1_input_CreateFcn 设置编辑框的初始值为"120"，函数代码如下:
function f1 input CreateFcn(hObject, eventdata, handles)
% hObject    handle to f1 input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
set(gcbo,'String','120')
if ispc && isequal
(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
       set(hObject,'BackgroundColor','white');
end
```

调用 f2_input_CreateFcn 设置编辑框的初始值为“60”，函数代码如下：

```
function f2_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to f2_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
set(gcbo,'String','60')
if ispc && isequal
(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end
```

调用 t_input_CreateFcn 设置编辑框的初始值为“0:0.001:0.25”，函数代码如下：

```
function t_input_CreateFcn(hObject, eventdata, handles)
% hObject    handle to t_input (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called
% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
set(gcbo,'String','0:0.001:0.25')
if ispc && isequal
(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
        set(hObject,'BackgroundColor','white');
end
```

调用 close_menu_Callback 结束程序，函数代码如下：

```
function close_menu_Callback(hObject, eventdata, handles)
% hObject    handle to close_menu (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close
```

在上述程序中，f1_input_CreateFcn、f2_input_CreateFcn 和 t_input_CreateFcn 在创建图形时依次给三个编辑文本框输入参数，plot_pushbutton_Callback 执行绘制图形的功能。

运行程序后，单击“Plot”按钮绘制图形，如图 7-53 所示。

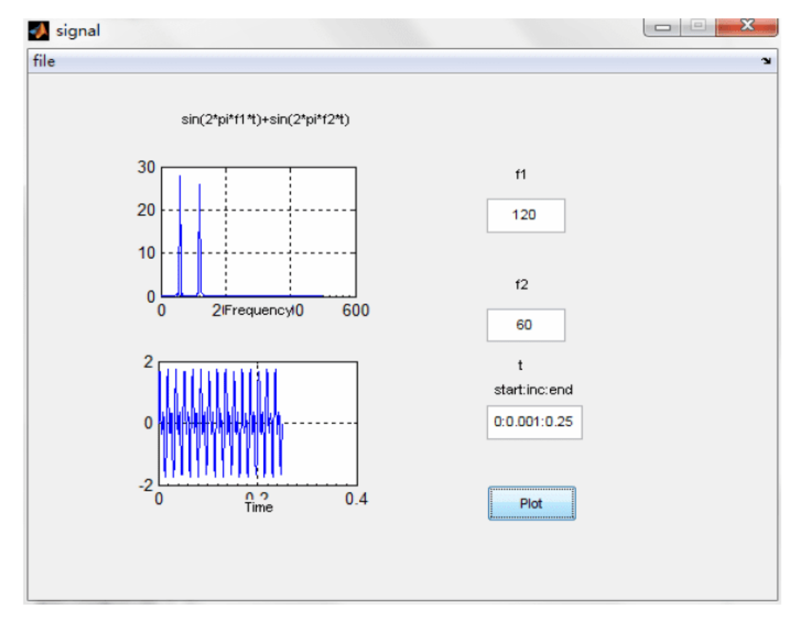


图 7-53　示例运行界面 1

可以在编辑框中改变频率和时间的数值，单击【file】/【plot】菜单命令，绘制的图形如图 7-54 所示。

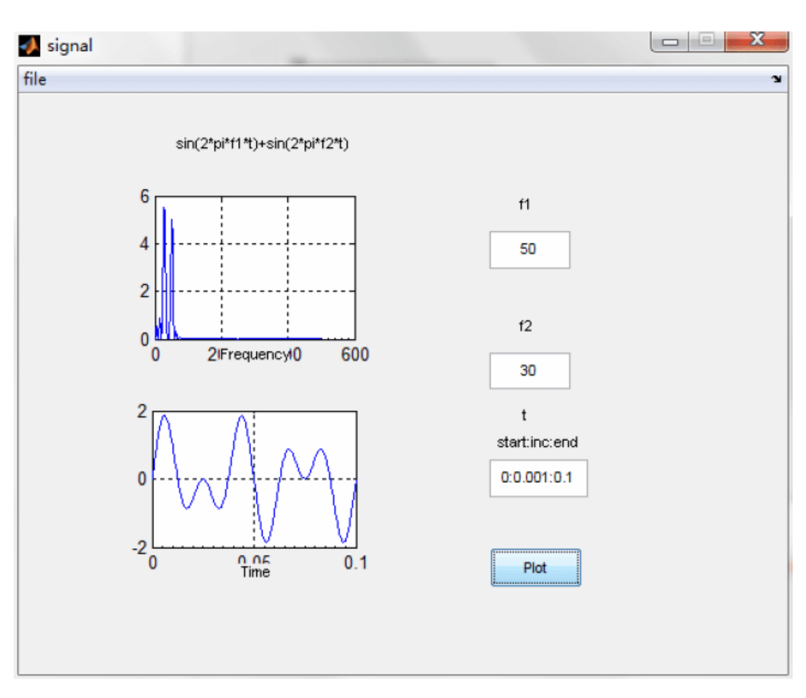


图 7-54　示例运行界面 2

7.4　本章小结

本章主要介绍图形用户界面的组成、程序设计和应用。在图形用户界面组成部分，介绍了组成图形用户界面的窗口、菜单、按钮及文字说明等各种对象。在程序设计部分，介绍了通过对控件的 Callback 属性编程方法。在图形用户界面的应用部分，给出了图形用户界面设计的一般步骤，并通过一个实例给出了图形用户界面所对应的完整 M 文件。

7.5　习题

（1）设计实现三个按钮的交互。

（2）设计实现 GUI 与工作空间里的数据交互。

（3）GUI 编程设计：使用 GUIDE 设计界面，分析正弦信号，并绘制相应的余弦、正切、余切的三角函数图形。

第 8 章　MATLAB 科学计算

本章主要介绍经常用到的 MATLAB 科学计算问题的求解方法，其中包括线性方程与非线性方程及常微分方程的求解、数据统计处理、数据插值、数值积分及优化问题求解等方面的内容。

8.1　方程求解

本节将分别讨论线性方程组、非线性方程组和常微分方程三种常见方程的解法。

8.1.1　线性方程组

线性方程组是线性代数中的主要内容之一，也是理论发展最为完整的部分。在 MATLAB 中包含多种处理线性方程组的命令，下面详细进行介绍。

1. 问题描述

在实际应用中，经常需要求解如下所示的两类线性方程组，其中第一种更常见。

AX=B

XA=B

按照数学的严格定义，并没有矩阵除法的概念，而 MATLAB 为了书写简便提供了用除号求解线性方程组解的方式，其具体用法如下。

- **X=A\B**　左除，计算方程组 AX=B 的解。
- **X=B/A**　右除，计算方程组 XA=B 的解。

下面针对 AX=B 的形式进行说明。系数矩阵 A 是 $m\times n$ 的矩阵，根据其维数可以分为如下 3 种情况。

- $m=n$ 为恰定方程组，即方程数等于未知量数。
- $m>n$ 为超定方程组，即方程数大于未知量数。
- $m<n$ 为欠定方程组，即方程数小于未知量数。

线性方程组解的类型也可以分为 3 种情况：

- rank(A)= rank([A, B])且对应齐次方程组 AX=0 不存在非 0 解，则方程组有唯一解，如矩阵 A 可逆。
- rank(A)= rank([A, B])且对应齐次方程组 AX=0 存在非 0 解，则方程组有无穷解，如 rank(A)= rank([A, B])且为欠定方程组。
- rank(A)= rank([A, B])，则方程组无解。

不难看出，线性方程组解的类型是由对应齐次方程组的解，对应系数矩阵和增广矩阵间的关系决定的。

2. 解的形式

线性方程组 AX=B 解的形式可以如下描述。

- ❑ 首先可以使用 null 函数求解对应齐次方程组 AX=0 的基础解系，也可以称为通解，则 AX=B 的解都可以通过通解的线性组合表示。
- ❑ 其次求解非齐次线性方程组 AX=B 的特解。
- ❑ 最后非齐次线性方程组 AX=B 解的形式为通解的线性组合加上特解。

3. 除法及求逆的解法

1）除法解法

若线性方程组 AX=B 的系数矩阵可逆，则 A\B 给出方程组的唯一解。

【例 8-1】 使用除法求解系数矩阵可逆的恰定线性方程组。

在命令窗口中输入如下语句。

```
A=pascal(4)
det A=det(A)
B=rand(4,1)
X1=A\B
X2=inv(A)*B
```

命令窗口中的输出结果如下所示。

```
A =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
det_A =
     1
B =
    0.8147
    0.9058
    0.1270
    0.9134
X1 =
   -2.5813
    9.1360
   -8.1751
    2.4351
X2 =
   -2.5813
    9.1360
   -8.1751
    2.4351
```

本例需要说明的是，det(A)用于计算矩阵 A 的行列式，不难看出矩阵 A 可逆；由于矩阵 A 是 4×4，所以矩阵 B 必须是 4×1，例中随机生成矩阵的语句是 B=rand(4,1)，而非 B=rand(1,4)；A\B 等价于 inv(A)*B。

若线性方程组 AX=B 的系数矩阵不可逆，则方程组的解不存在或不唯一，此时，执行

A\B 可能给出警告信息和不恰当的解。

【例 8-2】 使用除法求解系数矩阵不可逆的恰定线性方程组。

在命令窗口中输入如下语句。

```
A=[1 3 7;-1 4 4;1 10 18];
det_A=det(A)
B=[6;4;15]
X=A\B
```

命令窗口中的输出结果如下所示。

```
det A =
     0
B =
     6
     4
    15
Warning: Matrix is singular to working precision.
X =
   NaN
   Inf
  -Inf
```

从以上的结果可以看出，MATLAB 会显示提示信息，表示该矩阵是奇异矩阵，因此无法得到精确的数值解。

【例 8-3】 使用除法求解欠定线性方程组。

在命令窗口中输入如下语句。

```
C=magic(4);
A=C(1:3,:)
B=[1;0;0];
X=A\B
```

命令窗口中的输出结果如下所示。

```
A =
    16     2     3    13
     5    11    10     8
     9     7     6    12
X =
    0.1863
    0.0294
         0
   -0.1569
```

【例 8-4】 使用除法求解超定线性方程组。

在命令窗口中输入如下语句。

```
T=magic(5)
A=T(:,1:4)
```

```
B=[1;0;0;0;0];
X=A\B
```

命令窗口中的输出结果如下所示。

```
T =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9
A =
    17    24     1     8
    23     5     7    14
     4     6    13    20
    10    12    19    21
    11    18    25     2
X =
   -0.0041
    0.0437
   -0.0305
    0.0060
```

2）求解逆法

在例 8-1 中，已经介绍了通过求逆的方法求解方程组的解，这里着重介绍伪逆的用法。对于系数矩阵而言，它可能是方阵但不可逆，也可能不是方阵，上述情况都导致它的逆不存在或无定义，这就需要引入伪逆的概念。伪逆包含很多种形式（详见矩阵的有关书籍），下面介绍最常用的基于最小二乘意义下的最优伪逆，在 MATLAB 中通过 pinv 函数可以实现，即可以使用矩阵 A 的伪逆矩阵 pinv(A)来得到方程的一个解，其对应的数值解为 pinv(A)*B。

【例 8-5】 使用伪逆矩阵的方法求解奇异矩阵的线性方程的解。

在命令窗口中输入如下语句。

```
A=[1 3 7;-1 4 4;1 10 18];
B=[5;2;12];
X=pinv(A)*B
C=A*X
```

命令窗口中的输出结果如下所示。

```
X =
     0.3850
    -0.1103
     0.7066
C =
    5.0000
    2.0000
   12.0000
```

从上面的结果可以看出，通过使用伪逆矩阵的方法，可以求解得到数值解，同时该数

值解可以精确地满足结果。

上面都是介绍如何计算特解，下面介绍如何计算线性方程组的所有解。

【例 8-6】 使用求逆法计算线性方程组的所有解。

在命令窗口中输入如下语句。

```
A=[1 2 3 4;5 6 7 8;9 10 11 12];
B=[1;1;2];
X1=null(A)
X2=pinv(A)*B
```

命令窗口中的输出结果如下所示。

```
X1 =
    0.5135    0.1906
   -0.8267    0.1287
    0.1129   -0.8290
    0.2003    0.5098
X2 =
   -0.1250
   -0.0208
    0.0833
    0.1875
```

此时线性方程组的所有解为 X=a*X1(:,1)+b*X1(:,2)+X2，其中 a、b 为任意实数。

4. 矩阵分解的解法

1）LU 分解

LU 分解又可称为 Gauss（高斯）消去法。若系数矩阵为方阵，它可以表示为下三角矩阵和上三角矩阵的乘积，即 A=LU，其中，L 为下三角阵，U 为上三角阵。在 MATLAB 中通过 lu 函数可实现 LU 分解。

针对 LU 分解，线性方程组 AX=B 可以表示为 LUX=B，由于 L 和 U 的特殊性，通过 X=U/(L/B)求解可以大大提高运算速度。

LU 函数的具体语法形式如下。

- ❑ **[L,U]=lu(X)** X 是任意方阵，L 是下三角阵，U 是上三角阵，这三个变量满足的条件式为 X=LU。
- ❑ **[L,U,P]=lu(X)** X 是任意方阵，L 是下三角阵，U 是上三角阵，P 是置换矩阵，满足的条件式为 PX=LU。
- ❑ **Y=lu(X)** X 是任意方阵，把上三角矩阵和下三角矩阵合并在矩阵 Y 中给出，满足的条件式为 Y=L+U−I，但是将损失置换矩阵 P 的信息。

【例 8-7】 使用 LU 分解法求解线性方程组 $\begin{cases} 2x_1+x_2+x_3=1 \\ x_1-x_2+3x_3=5 \\ 6x_1-5x_2+x_3=7 \end{cases}$。

在命令窗口中输入如下语句。

```
A=[2 1 1;1 -2 3;6 -5 1];
B=[1;5;7];
```

```
C=det(A)
[L,U]=lu(A)
X=U\(L\B)
```

命令窗口中的输出结果如下所示。

```
C =
    50
L =
    0.3333    1.0000         0
    0.1667   -0.4375    1.0000
    1.0000         0         0
U =
    6.0000   -5.0000    1.0000
         0    2.6667    0.6667
         0         0    3.1250
X =
    0.3600
   -0.7600
    1.0400
```

【**例 8-8**】 使用 LU 分解法求解线性方程组 $\begin{cases}-x_1+8x_2+5x_3=2\\9x_1-x_2+2x_3=3\\2x_1-5x_2+7x_3=5\end{cases}$ 。

在命令窗口中输入如下语句。

```
A=[-1 8 5;9 -1 2;2 -5 7];
B=[2;3;5];
C=det(A)
[L,U]=lu(A)
X=U\(L\B)
```

命令窗口中的输出结果如下所示。

```
C =
  -690
L =
   -0.1111    1.0000         0
    1.0000         0         0
    0.2222   -0.6056    1.0000
U =
    9.0000   -1.0000    2.0000
         0    7.8889    5.2222
         0         0    9.7183
X =
    0.1913
   -0.0957
    0.5913
```

2）Cholesky 分解

若系数矩阵为对称正定矩阵，可以表示为上三角矩阵和其转置的乘积，即 A=R'R，其

中 R 为上三角矩阵，R'为下三角矩阵。在 MATLAB 中通过 chol 函数可以实现 Cholesky 分解。

从理论角度来讲，并不是所有的对称矩阵都可以进行 Cholesky 分解，可以进行 Cholesky 分解的矩阵必须是正定的。针对 Cholesky 分解，线性方程组 AX=B 可以表示为 R'RX=B，由于 R 的特殊性，通过 X=R/(R'/B')求解可以大大提高运算速度。

chol 函数的具体语法形式如下。

- **R=chol(X)**　X 是对称的正定矩阵，R 是上三角矩阵，使得 X=R'R。如果矩阵 X 是非正定矩阵，该命令会返回错误信息。
- **[R,p]=chol(X)**　该命令返回两个参数，并不返回错误信息。当 X 是正定矩阵时，返回的矩阵 R 是上三角矩阵，而且满足 X=R'R，同时返回参数 p=0；当 X 不是正定矩阵时，返回的参数 p 是正整数，R 是三角矩阵，矩阵阶数是 p-1，并且满足 X=X(1:p-1,1:p-1)=R'R。

【例 8-9】使用 Cholesky 分解法求解线性方程组 $\begin{cases} 2x_1 + x_2 + x_3 = 1 \\ x_1 - 5x_2 + 4x_3 = 3 \\ 3x_1 - 4x_2 + 6x_3 = 5 \end{cases}$ 。

在命令窗口中输入如下语句。

```
A=[2 1 3;1 5 4;3 4 6];
B=[1;3;5];
C=det(A)
R=chol(A)
TR=R'
X=R\(TR\B)
```

命令窗口中的输出结果如下所示。

```
C =
     1
R =
    1.4142    0.7071    2.1213
         0    2.1213    1.1785
         0         0    0.3333
TR =
    1.4142         0         0
    0.7071    2.1213         0
    2.1213    1.1785    0.3333
X =
  -23.0000
  -10.0000
   19.0000
```

【例 8-10】 对对称正定矩阵进行 Cholesky 分解。

在命令窗口中输入如下语句。

```
n=5;
X=pascal(n)
```

```
R=chol(X)
C=transpose(R)*R
```

命令窗口中的输出结果如下所示。

```
X =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
R =
     1     1     1     1     1
     0     1     2     3     4
     0     0     1     3     6
     0     0     0     1     4
     0     0     0     0     1
C =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    70
```

由结果可以看出，R 是上三角矩阵，同时满足 X=R'R=C，表明上面的 Cholesky 分解过程是正确的。

如果修改矩阵的信息，在命令窗口中继续输入如下语句。

```
X(n,n)=X(n,n)-1
[R1,p]=chol(X)
C1=transpose(R1)*R1
C2=X(1:p-1,1:p-1)
```

命令窗口中的输出结果如下所示。

```
X =
     1     1     1     1     1
     1     2     3     4     5
     1     3     6    10    15
     1     4    10    20    35
     1     5    15    35    69
R1 =
     1     1     1     1
     0     1     2     3
     0     0     1     3
     0     0     0     1
p =
     5
C1 =
     1     1     1     1
     1     2     3     4
```

```
     1     3     6    10
     1     4    10    20
C2 =
     1     1     1     1
     1     2     3     4
     1     3     6    10
     1     4    10    20
```

由此可见，当原来的正定矩阵的最后一个元素减 1 后，矩阵将不是正定矩阵，并且满足 X(1:p-1,1:p-1)=R'R。

（3）QR 分解

对于任何系数矩阵，可以表示为正交矩阵和上三角矩阵的乘积，即 A=QR，其中 Q 为正交矩阵，R 为上三角矩阵。在 MATLAB 中通过 qr 函数可以实现 QR 分解。

针对 QR 分解，线性方程组 AX=B 可以表示为 QRX=B，由于 Q 和 R 的特殊性，通过 X=R/(Q/B)求解可以大大提高运算速度。

Qr 函数的具体语法形式如下。

[Q,R]=qr(A)：矩阵 R 和矩阵 A 的大小相同，Q 是正交矩阵，满足 A=QR，该调用方式适合于满矩阵和稀疏矩阵。

【例 8-11】 使用 QR 分解法求解线性方程组 $\begin{cases} 2x_1 + x_2 + 3x_3 + 4x_4 = 1 \\ x_1 + 5x_2 + 4x_3 + 2x_4 = 3 \\ 3x_1 + 4x_2 + 6x_3 - 5x_4 = 5 \end{cases}$ 。

在命令窗口中输入如下语句。

```
A=[2 1 3 4;1 5 4 2;3 4 6 -5];
B=[1;3;5];
[Q,R]=qr(A)
X=R\(Q\B)
```

命令窗口中的输出结果如下所示。

```
Q =
   -0.5345    0.4257   -0.7301
   -0.2673   -0.9047   -0.3319
   -0.8018    0.0177    0.5974
R =
   -3.7417   -5.0780   -7.4833    1.3363
         0   -4.0267   -2.2351   -0.1951
         0         0    0.0664   -6.5709
X =
         0
    0.2846
    0.4878
   -0.1870
```

5. 共轭梯度的解法

MATLAB 还提供了一系列的线性方程组 AX=B 的共轭梯度解法，这里主要介绍双共

轭梯度法，它可由 bicg 函数实现。bicg 函数要求系数矩阵 A 必须为方阵。

【例 8-12】 使用共轭梯度法求解线性方程组 $\begin{cases}5x_1+3x_2+x_3=7\\2x_1-3x_2+4x_3=9\\x_1-7x_2+2x_3=1\end{cases}$。

在命令窗口中输入如下语句。

```
A=[3 1 1;2 -1 5;8 -4 0];
B=[2;4;1];
[X, flag,relres,iter,resvec]=bicg(A,B)
```

命令窗口中的输出结果如下所示。

```
X =
    0.3000
    0.3500
    0.7500
flag =
     0
relres =
     5.4820e-016
iter =
     3
resvec =
    4.5826
    3.9370
    6.6052
    0.0000
```

其中，flag 表示在默认迭代次数内收敛，relres 表示相对残差 norm(B-A*x)/norm(B)，iter 表示终止的迭代次数，resvec 表示每次迭代的残差。

8.1.2 非线性方程

1. 函数的零点

对于任意函数，在求解范围内可能有零点，也可能没有；可能只有一个零点，也可能有多个甚至无数个零点。MATLAB 没有可以求解所以函数零点的通用命令，本节将分别讨论一元函数和多元函数的零点求解问题。

1）一元函数的零点

在所有函数中，一元函数是最简单的，在 MATLAB 中可以使用 fzero 函数计算一元函数的零点，其具体使用方法如下。

- ❑ **x=fzero(fun,x0)** 在 x0 点附近寻找函数 fun 的零点。
- ❑ **x=fzero(fun,x0,options)** 使用 options 设定优化器参数。
- ❑ **x=fzero(fun,[x0,x1])** 在[x0,x1]区间寻找函数 fun 的零点。

【例 8-13】 计算一元函数 $f(x)=x^2\sin x-x+1$ 在[-3,4]区间的零点。

首先绘制函数的曲线，在命令窗口中输入如下语句。

```
x=-3:0.1:4;
y=x.*x.*sin(x)-x+1;
plot(x,y, 'r')
xlabel('x');
ylabel('f(x)');
title('The zero of function')
hold on
h=line([-3,4],[0,0]);
set(h,'color','g')
grid;
```

图形窗口中的输出结果如图 8-1 所示。

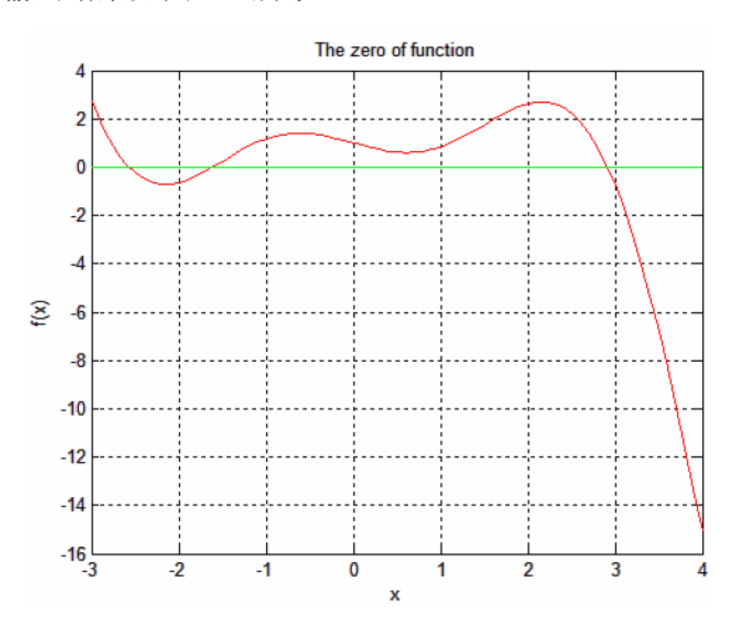


图 8-1　一元函数曲线

在求解函数零点之前，需要绘制函数的图形，是为了在后面的步骤中使用 fzero 命令时，更加方便地选择初始数值 x0。

由图 8-1 不难看出，曲线在[-3,4]区间内包含 3 个零点。

其次计算函数某点附近的零点，在命令窗口中输入如下语句。

```
f=@(x)x-10*sin(x);
X1=fzero(f,-2.5)
X2=fzero(f,-1.5)
X3=fzero(f,3)
```

命令窗口中的输出结果如下所示。

```
X1 =
   -2.8523
X2 =
   -2.8523
X3 =
    2.8523
```

求一元函数零点时，可以使用 optimset 函数设置优化器参数，其具体使用方法如下。

- ❑ **optimset** 显示优化器的现有参数名及其参数值。
- ❑ **options=optimset('param1',value1, 'param2',value2,...)** 使用参数名和参数值设定优化器的参数。
- ❑ **options=optimset(oldopts, 'param1',value1,...)** 在现有优化器 oldopts 的基础上，使用参数名和参数值变更优化器参数。

optimset 函数中可以设置的主要优化器参数如表 8-1 所示。

表 8-1 优化器参数

参 数 名	有效参数值	功 能 描 述
Display	final、off、liter notify	final：只显示最终结果，该选项为默认值 off：不显示计算结果 liter：显示每个迭代步骤的计算结果 notify：只在不收敛时显示计算结果
MaxFunEvals	正整数	最大允许的函数评估次数
MaxIter	正整数	最大允许的迭代次数
TolFun	正标量	函数值的截断阈值
TolX	正标量	自变量的截断阈值
OutputFcn	空矩阵或者用户定义函数句柄	空矩阵：迭代过程采用 MATLAB 自带的函数 用户自定义函数句柄：用该函数替换 MATLAB 自带的函数
FunValCheck	off 和 on	off：不检查输入函数的返回值，该选项为默认值 on：如果输入函数的返回值为复数或者 NaN，则显示警告信息

2）多元函数的零点

多元函数的零点问题比一元函数的零点问题更难解决，但是当零点大致位置和性质比较好预测时，也可以使用数值方法来搜索精确的零点。

非线性方程组的标准形式为 $F(x)=0$，其中 x 为向量，$F(x)$为函数向量。在 MATLAB 中，求解多元函数的命令是 fsolve，它的具体使用方法如下。

- ❑ **X=fsolve(fun,x0)** 在向量 x0 附近寻找函数 fun 的解。
- ❑ **X=fsolve(fun,x0,options)** 使用 options 设定优化器参数。

其中 options 设定优化器参数的方法同前。

【例 8-14】 求解二元方程组 $\begin{cases} 2x_1 - x_2 = e^{-x_1} \\ -x_1 + 2x_2 = e^{-x_2} \end{cases}$ 的零点。

首先绘制函数的曲线，在命令窗口中输入如下语句。

```
x=[-5:0.1:5];
y=x;
[X,Y]=meshgrid(x,y);
Z=2*X-Y-exp(-X);
surf(X,Y,Z)
xlabel('x')
xlabel('y')
```

```
xlabel('z')
title('The figure of the function')
```

图形窗口中的输出结果如图 8-2 所示。

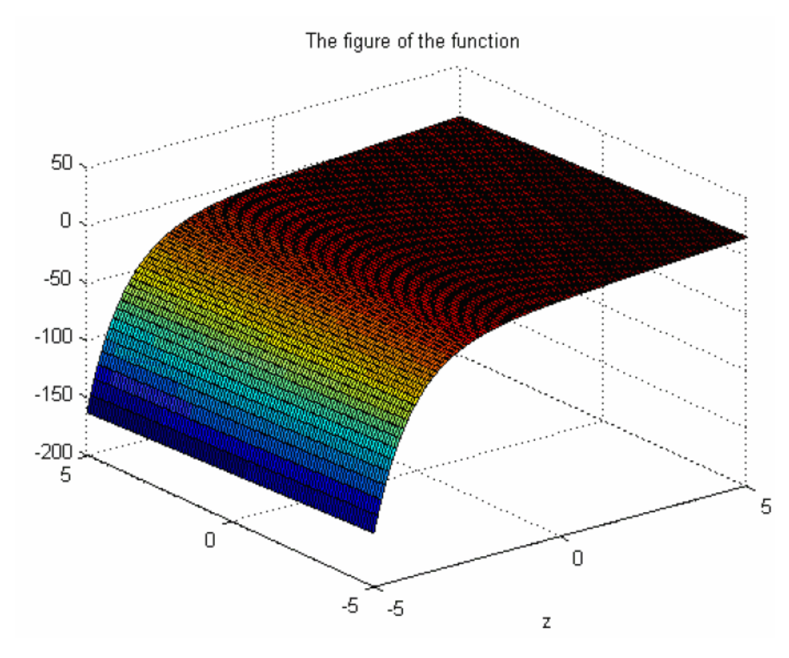


图 8-2　二元方程组曲线

编写求解函数的 M 文件，在其中输入如下的程序代码。

```
function F=fsolvefun(x)
F=[2*x(1)- x(2)-exp(-x(1));-x(1)+2*x(2)-exp(-x(2))];
```

将上述程序代码保存为 fsolvefun.m 文件。

下面求解二元函数的零点，在命令窗口中输入如下语句。

```
x0=[-5;-5];
options=optimset('Display','iter');
x=fsolve(@fsolvefun,x0,options)
```

命令窗口中的输出结果如下所示。

```
Norm of      First-order   Trust-region
 Iteration  Func-count      f(x)          step          optimality    radius
     0          3          47071.2                       2.29e+004         1
     1          6          12003.4              1        5.75e+003         1
     2          9          3147.02              1        1.47e+003         1
     3         12          854.452              1              388         1
     4         15          239.527              1              107         1
     5         18          67.0412              1             30.8         1
     6         21          16.7042              1             9.05         1
     7         24          2.42788              1             2.26         1
     8         27          0.032658      0.759511            0.206       2.5
     9         30     7.03149e-006       0.111927          0.00294       2.5
    10         33     3.29525e-013     0.00169132        6.36e-007       2.5
Optimization terminated: first-order optimality is less than options.TolFun.
x =
```

```
    0.5671
    0.5671
```

由上面的结果可以看出，原来的二元函数是对称的，因此，求解的未知数结果是相等的。

2. 非线性方程组的解

求非线性方程组的解的问题，也就是求多元函数的零点问题。在MATLAB中使用fsolve函数计算非线性方程组的解，使用方法见上一小节。

【例 8-15】 求解 $X^4=\begin{bmatrix}1 & 7\\ -11 & 9\end{bmatrix}$。

首先对非线性方程组进行函数描述，并保存为myfun8_15.m，其内容如下所示。

```
function T=myfun8_15(x)
T=x^4-[1,7;-11,9];
```

其次对非线性方程组进行求解，在命令窗口中输入如下语句。

```
x0=[3 1;2 1];
options=optimset('Display','off');
X=fsolve(@myfun8_15,x0,options)
```

命令窗口中的输出结果如下所示。

```
X =
    1.4693    0.3875
   -0.6089    1.9121
```

8.1.3 常微分方程

1. 求解常微分方程的函数

在MATLAB中使用ode45、ode23、ode113、ode15s、ode23s、ode23t、ode23tb等函数求常微分方程（ODE）的数值解，这些函数的介绍如表8-2所示。其具体使用方法类似，为了方便后面的描述，这里用solver统一代替它们。

表 8-2 常微分方程的求解算法

算 法 名	含 义	特 点	说 明
ode23	普通2-3阶法非刚性解	一步法：2、3阶Runge-Kutta方程；累计截断误差达 $(\Delta x)^3$	适用于精度较低的情形
ode23s	低阶法解刚性	一步法：2阶Rosebrock算法；低精度	当精度较低时，计算时间比ode15s短
ode23t	解适度刚性	梯形算法	适用于刚性情形
ode23tb	低阶法解刚性	梯形算法：低精度	当精度较低时，计算时间比ode15s短
ode45	普通4-5阶法非刚性解	一步算法：4、5阶Runge-Kutta方程；累计截断误差达 $(\Delta x)^3$	大部分场合的首选算法

续表

算 法 名	含 义	特 点	说 明
ode15s	变阶法解刚性	多步法：Gear's 反向数值微分；精度中等	若 ode45 失效，可尝试使用
ode113	普通变阶法非刚性解	多步法：Adams 算法；高低精度均可达到	计算时间比 ode45 短

在介绍具体用法之前，首先介绍其涉及的参数，如表 8-3 所示。

表 8-3　solver 中的参数

参 数 名	功 能 描 述
odefun	表示常微分方程
tspan	表示求解区间或求解时刻，通常为 tspan=[t0,tf]或 tspan=[t0,t1,t2,…,tf](要求单调)
y0	表示初始条件
options	表示使用 odeset 函数所设置的可选参数
p1,p2	表示传递给 odefun 的参数

其次介绍它们的具体用法，如下所述。

```
[T,Y]=solver(odefun,tspan,y0,options)
```

在区间 tspan=[t0,tf]上，从 t0 到 tf，用初始条件 y0 求解显示微分方程 y=f(t,y)。对于标量 t 和列向量 y，函数 f=odefun(t,y)必须返回 f(t,y)的列向量 f。解向量 Y 中的每行结果对应于时间向量 T 中每个时间点。

要获得在其他指定时间点 t0,t1,t2,…,tf 上的解，则令 tspan=[t0,t1,t2,…,tf]（要求是单调的）。

利用 odeset 函数所设置的可选参数进行求解，odeset 函数可设置的参数如表 8-4 所示，其用法类似于 optimset 函数。

表 8-4　solver 中 options 的参数

参 数 名	取 值	含 义
absTol	有效值：正实数或向量 1e-6	绝对误差对应于解向量中的所有元素；向量则分别对应于解向量默认值中的每一分量
relTol	有效值：正实数 默认值：1e-6	相对误差对应于解向量中的所有元素。在每步（第 k 步）计算过程中，误差估计为 e(k)<=max(RelTol*abs(y(k)),AbsTol(k))
events	有效值：on、off	为 on 时，返回相应的事件记录
normControl	有效值：on、off 默认值：off	为 on 时，控制解向量范数的相对误差，使每步计算中，满足 norm(e)<=max(RelTol*norm(y),AbsTol)
outputFcn	有效值：odeplot、odephas2、odephas3、odeprint 默认值：odeplot	若无输出参量，则 solver 将执行下面操作之一：画出解向量中各元素随时间的变化 画出解向量中前两个分量构成的相平面图 画出解向量中前三个分量构成的三维相空间图 随计算过程，显示解向量
outputSel	有效值：正整数或向量 默认值：[]	若不使用默认设置，则 OutputFcn 所表现的是那些正整数指定的解向量中的分量的曲线或数据
refine	有效值：正整数或 k>1 默认值：k=1	若 k>1，则增加每个积分步中的数据点记录，使解曲线更加光滑
jacobian	有效值：on、off 默认值：off	若为 on，返回相应的 ode 函数的 jacobi 矩阵

续表

参 数 名	取 值	含 义
jpattern	有效值：on、off 默认值：off	为 on 时，返回相应的 ode 函数的稀疏 jacobi 矩阵
mass	有效值：none、M、M(t)、M(t,y) 默认值：none	M：不随时间变化的常数矩阵 M(t)：随时间变化的矩阵 M(t,y)：随时间、地点变化的矩阵
maxStep	有效值：正实数 默认值：tspans/0	最大积分步长

```
[T,Y]=solver(odefun,tspan,y0,options,p1,p2,…)
```

利用传递给函数 odefun 的 p1,p2,…等参数进行求解。

2. 求解常微分方程的类型

MATLAB 可以求解 3 种一阶常微分方程，即显式常微分方程、线性隐式常微分方程和完全隐式常微分方程。

显式常微分方程的形式如 $\begin{cases} y'=f(t,y) \\ y(t_0)=y_0 \end{cases}$。

线性隐式常微分方程的形式如 $\begin{cases} M(t,y)y'=f(t,y) \\ y(t_0)=y_0 \end{cases}$。

完全隐式常微分方程的形式如 $\begin{cases} f(t,y,y')=0 \\ y(t_0)=y_0 \end{cases}$。

对于高阶常微分方 $y^{(n)}=f(t,y,y',...,y^{(n-1)})$，可以将其转换成如下所示的一阶常微分方程组 $\begin{cases} y_1'=y_2 \\ y_2'=y_2 \\ . \\ . \\ y_n'=f(t,y_1,y_2,...y_n) \end{cases}$。

3. 具体解法

下面通过 3 个实例分别说明 3 种方程的解法。

【例 8-16】 已知微分方程 $y''-\mu(1-y^2)y'+y=0(y(0)=0,y'(0)=2;t\in[0,30])$，该方程为显式常微分方程，分别取 $\mu=3$ 和 $\mu=5$ 求解该方程。

首先对微分方程进行变换得到如下形式：$\begin{cases} y_1'=y_2 \\ y_2'=\mu(1-y_1^2)y_2-y_1 \end{cases}$

其次对方程组进行函数描述，并保存为 myfun8_16.m，其内容如下所示。

```
function output=myfun8_16(t,y,mu)
output=zeros(2,1);
output(1)=y(2);
output(2)=mu*(1-y(1)^2)*y(2)-y(1);
```

再次对方程组进行求解，在命令窗口中输入如下语句。

```
[t1,y1]=ode45(@myfun8 16,[0 30],[0;2],[],3);  %mu=3
[t2,y2]=ode45(@myfun8 16,[0 30],[0;2],[],5);  %mu=5
plot(t1,y1(:,1), '-',t2,y2(:,2),'--')
title('显式常微分方程的解');
xlabel('t');
ylabel('y');
legend('mu=3', 'mu=5');
```

图形窗口中的输出结果如图 8-3 所示。

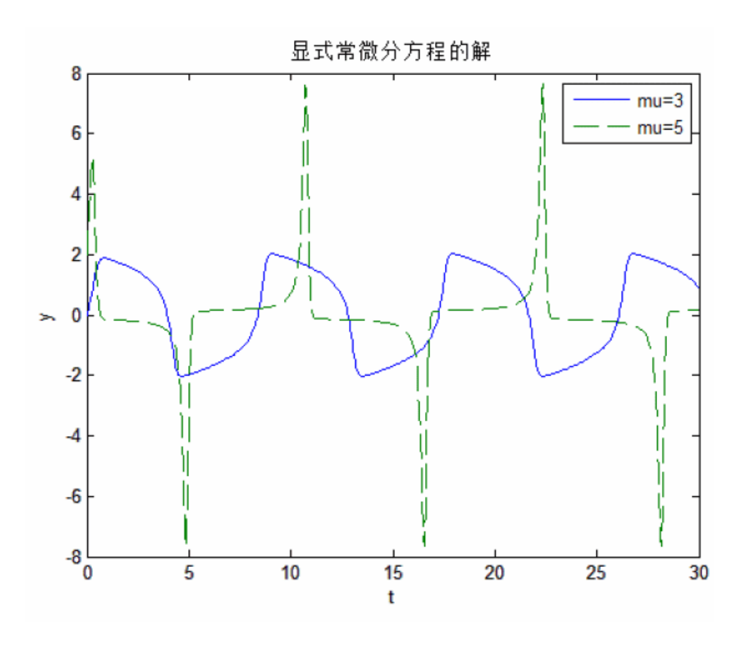


图 8-3　显式常微分方程的解

【例 8-17】 求解微分方程 $(ty^2+1)y'=3y^3+y+4(t\in[0,10];y(0)=2)$，该方程为线性隐式常微分方程。

首先根据微分方程 $(ty^2+1)y'=3y^3+y+4$ 和通式 $M(t,y)y'=f(t,y)$，得到

$$\begin{cases} f(t,y)= & 3y^3+y+4 \\ M(t,y)= & ty^2+1 \end{cases}$$

其次对 $f(t,y)$ 进行函数描述，并保存为 myfun8_17f.m，其内容如下所示。

```
function output=myfun8 17f(t,y)
output=3*y.^3+y+4;
```

再次对 $\mathrm{M}(t,y)$ 进行函数描述，并保存为 myfun8_17M.m，其内容如下所示。

```
function output=myfun8 17M(t,y)
output=t*y.^2+1;
```

最后对方程进行求解，在命令窗口中输入如下语句。

```
options=odeset('RelTol',1e-6, 'OutputFcn', 'odeplot', 'Mass',@myfun8_17M);
[t,y]=ode45(@myfun8_17f,[0 10],2,options);
xlabel('t');
ylabel('y');
title('线性隐式常微分方程的解')
```

图形窗口中的输出结果如图 8-4 所示。

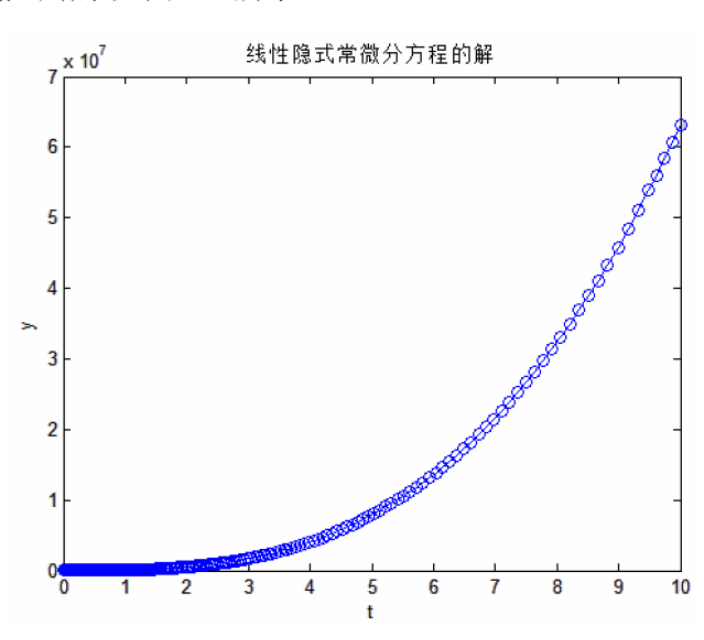


图 8-4　线性隐式常微分方程的解

MATLAB 提供了函数 decic 计算自洽的初始值，其具体用法如下。

[t,Y]=ode15i(odefun,tspan,y0,yp0,options)：y0 和 yp0 用于指定微分方程的初始值，初始值必须是自洽的，即满足 f(t,y0,yp0)=0。

MATLAB 提供了函数 decic 计算自洽的初始值，其具体用法如下。

[y0mod,yp0mod]=decic(odefun,t0,y0,fixed_y0,yp0,fixed_yp0,options)：t0 为初始时间，y0 和 yp0 是猜测的初始值，当 fixed_y0(i)=1 时，y0(i)不会被改变，fixed_yp0 的作用与 fixed_y0 相同。

【例 8-18】 求解微分方程 $ty^2(y')^3-2y^3(y')^2+3t(t^2+1)y'-t^2y=0(t\in[1,20];y(0)=\sqrt{3/2})$，该方程为完全隐式常微分方程。

首先对方程进行函数描述，并保存为 myfun8_18.m，其内容如下所示。

```
function output=myfun8_18(t,y,dydt)
output=t*y.^2*dydt.^3-2* y.^3*dydt.^2+3*t*(t^2+1)*dydt-t^2*y;
```

其次对方程进行求解，在命令窗口中输入如下语句。

```
t0=1;
y0=sqrt(3/2);
yp0=0;
[y0,yp0]=decic(@myfun8_18,t0,y0,1,yp0,0);
```

```
[t,y]=ode15i(@myfun8 18,[1 20],y0,yp0);
plot(t,y);
xlabel('t');
ylabel('y');
title('完全隐式常微分方程的解');
```

图形窗口中的输出结果如图 8-5 所示。

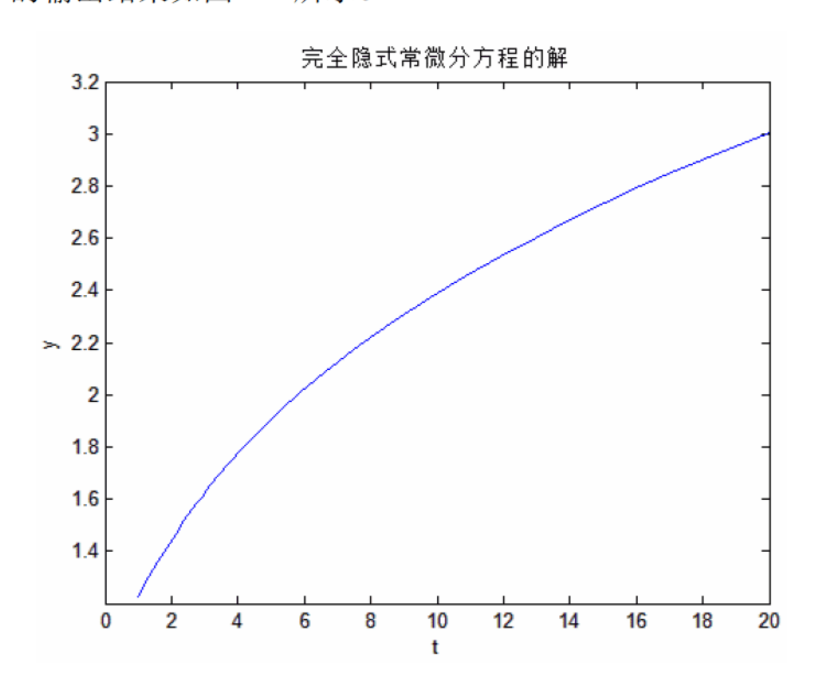


图 8-5　完全隐式常微分方程的解

8.2　数据处理统计

本节主要介绍 MATLAB 在数据统计处理方面的应用，包括最大值和最小值、求和和求积、平均值和中值、标准方差、相关系数以及排序等内容。

1. 随机数的生成

先来了解一下常见的随机数生成函数，如表 8-5 所示。

表 8-5　随机数生成函数

函　数　名	调 用 形 式	注　　释
unifrnd	unifrnd(A,B,m,n)	[A,B]上均匀分布（连续）随机数
unidrnd	unidrnd(N,m,n)	均匀分布（离散）随机数
maxStep	exprnd(Lambda,m,n)	参数为 Lambda 的指数分布随机数
uormrnd	normrnd(MU,SIGMA,m,n)	参数为 MU,SIGMA 的正态分布随机数
chi2rnd	chi2rnd(N,m,n)	自由度为 N 的卡方分布随机数
trnd	trnd(N,m,n)	自由度为 N 的 t 分布随机数

续表

函 数 名	调 用 形 式	注 释
frnd	frnd(N1,N2,m,n)	第一自由度为 N1，第二自由度为 N2 的 F 分布随机数
gamrnd	gamrnd(A,B,m,n)	参数为 A,B 的γ分布随机数
betarnd	betarnd(A,B,m,n)	参数为 A,B 的 β 分布随机数
lognrnd	lognrnd(MU,SIGMA,m,n)	参数为 MU,SIGMA 的对数正态分布随机数
nbinrnd	nbinrnd(R,P,m,n)	参数为 R,P 的负二项式分布随机数
ncfrnd	ncfrnd(N1,N2,delta,m,n)	参数为 N1,N2,delta 的非中心 F 分布随机数
nctrnd	nctrnd(N,delta,m,n)	参数为 N,delta 的非中心 t 分布随机数
ncx2rnd	ncx2rnd(N,delta,m,n)	参数为 N,delta 的非中心卡方分布随机数
ray1rnd	ray1rnd(B,m,n)	参数为 B 的瑞利分布随机数
weibrnd	weibrnd(A,B,m,n)	参数为 A,B 的韦伯分布随机数
binornd	binornd(N,P,m,n)	参数为 N,p 的二项分布随机数
geornd	geornd(P,m,n)	参数为 p 的几何分布随机数
hygernd	hygernd(M,K,N,m,n)	参数为 M,K,N 的超几何分布随机数
poissrnd	poissrnd(Lambda,m,n)	参数为 Lambda 的泊松分布随机数

【例 8-19】 生成[1 5]上均匀分布的 4×4 的随机数矩阵。

在命令窗口中输入如下语句。

```
x=unifrnd(1,5,4,4)
```

命令窗口中的输出结果如下所示。

```
x =
    3.5294    4.8300    4.8287    2.6870
    1.3902    4.8596    2.9415    4.6629
    2.1140    1.6305    4.2011    4.1688
    3.1875    4.8824    1.5675    4.8380
```

2. 数据分析基本函数

在 MATLAB 中提供了大量数据分析函数，在分类介绍这些函数之前，首先给出如下约定。

- ❑ 进行一维数据分析时，数据可以用行向量或列向量来表示，无论哪种表示方法，函数的运算都是对整个向量进行的。
- ❑ 进行二维数据分析时，数据可以用多个向量或二维矩阵来表示。对于二维矩阵，函数的运算总是按列进行的。

MATLAB 提供了包括计算随机变量数字特征在内的大量数据分析函数，详见以下小节内容。

8.2.1 最大值和最小值

在 MATLAB 中计算最大值和最小值的函数分别是 max 和 min，具体用法如下。

计算最大值函数：C=max(A)，如果 A 是向量，返回向量中的最大值；如果 A 是矩阵，

返回一个包含各列最大值的行向量。

计算最小值函数：C=max(A)，min 和 max 函数使用方法类似。

【例 8-20】 应用求最大值、最小值的函数。

在命令窗口中输入如下语句。

```
x=1:20;
y=randn(1,20);
figure;
hold on;
plot(x,y);
[y_max,I_max]=max(y)      %求向量最大值及其对应下标
plot(x(I max),y max, 'ro');
[y_min,I_min]=min(y)      %求向量最小值及其对应下标
plot(x(I_min),y_min, 'g*');
xlabel('x');
ylabel('y');
legend('原始数据', '最大值', '最小值');
```

命令窗口中的输出结果如下所示。

```
y max =
       1.5326
I max =
      16
y min =
      -1.2141
I_min =
      13
```

图形窗口中的输出结果如图 8-6 所示。

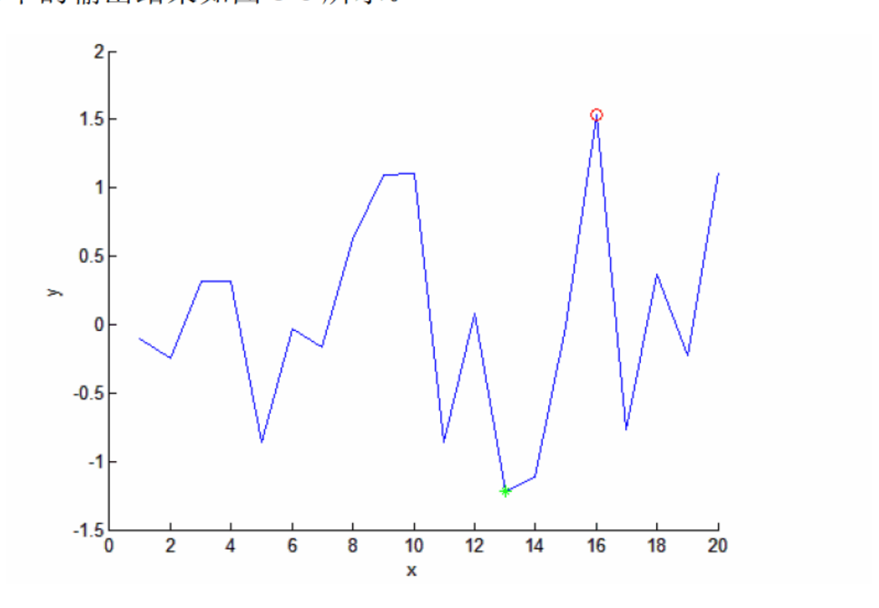


图 8-6　最大值与最小值

8.2.2 求和和求积

在 MATLAB 中求和和求积的函数分别为 sum 和 prod，具体用法如下。

计算元素和：B=sum(A)，如果 A 是向量，返回向量 A 的各元素之和；如果 A 是矩阵，返回含有各列元素之和的行向量。

计算元素连乘积：B=prod(A)，如果 A 是向量，返回向量 A 的各元素连乘积；如果 A 是矩阵，返回含有各列元素连乘积的行向量。

【例 8-21】 应用求和与求积的函数。

在命令窗口中输入如下语句。

```
x=1:20;
y=randn(1,20);
y_sum=sum(y)
y_prod=prod(y)
```

命令窗口中的输出结果如下所示。

```
y_sum =
   -1.3643
y_prod =
 -1.6120e-005
```

8.2.3 平均值和中值

在 MATLAB 中计算平均值和中值的函数分别为 mean 和 median，具体用法如下。

- **计算均值也叫数学期望** M=mean(A)，如果 A 是向量，返回向量 A 的平均值；如果 A 是矩阵，返回含有各列平均值的行向量。
- **计算中值** M=median(A)，与 mean 函数使用方法类似。

【例 8-22】 应用求平均值和中值的函数，在命令窗口中输入如下语句。

```
x=1:20;
y=randn(1,20);
y_mean=mean(y)              %求向量平均值
y_median=median(y)          %求向量中间值
```

命令窗口中的输出结果如下所示。

```
y_mean =
        0.0283
y_median =
        0.1461
```

8.2.4 标准方差

在 MATLAB 中求标准方差的函数为 std，具体用法如下。

s=std(A)：如果 A 是向量，返回向量的标准差；如果 A 是矩阵，返回含有各列标准差的行向量。

另外，MATLAB 计算方差（即标准差的平方）的函数为 var：

v=var(X)：如果 A 是向量，返回向量的方差；如果 A 是矩阵，返回含有各列方差的行向量。

向量 x 的标准差定义如下。

$$s=[\frac{1}{N-1}\sum_{k=1}^{N}(x_k-\overline{x})^2]^{\frac{1}{2}}$$

向量 x 的方差是标准差的平方，即

$$s^2=\frac{1}{N-1}\sum_{k=1}^{N}(x_k-\overline{x})^2$$

其中 N 是向量 x 的长度，$\overline{x}=\frac{1}{N}\sum_{k=1}^{N}x_k$，即平均值。

当 N 较大时，取 $s^2=\frac{1}{N}\sum_{k=1}^{N}(x_k-\overline{x})^2$，有时称此 s^2 为样本方差，而称上式的 s^2 为样本修正方差。

【例 8-23】 计算向量 x 的标准差。

在命令窗口中输入如下语句。

```
x=1:20;
mean x=mean(x);
r=0;
for i=1:20
   r=r+(x(i)-mean x)^2;
end
r1=sqrt(r/10)
r2=sqrt(r/9)
r3=std(x)
```

命令窗口中的输出结果如下所示。

```
r1 =
    8.1548
r2 =
    8.5959
r3 =
    5.9161
```

8.2.5 相关系数

MATLAB 提供了 corrcoef 函数计算相关系数，具体用法如下。

corrcoef(X,Y)：计算列向量 X,Y 的相关系数，等价于 corrcoef([X,Y])；corrcoef(A)计算矩阵 A 的列向量的相关系数矩阵。

另外，cov 函数用来计算协方差。

cov(X)计算向量 X 的协方差；cov(A)计算矩阵 A 各列的协方差矩阵，该协方差矩阵的对角线元素是 A 的各列的方差；cov(X,Y)等价于 cov([X,Y])。

【例 8-24】 计算协方差与相关系数。

在命令窗口中输入如下语句。

```
X=[1 0 1 3]';
Y=[-3 5 1 5]';
A1=cov(X)
A2=cov(X,Y)
A3=corrcoef(X)
A4= corrcoef(X,Y)
```

命令窗口中的输出结果如下所示。

```
A1 =
    1.5833
A2 =
    1.5833    1.0000
    1.0000   14.6667
A3 =
     1
A4 =
    1.0000    0.2075
    0.2075    1.0000
```

8.2.6 排序

在 MATLAB 中用函数 sort()实现数值的排序，具体用法如下。

B=sort(A)：如果 A 是向量，升序排列向量；如果 A 是矩阵，升序排列各个列。

B=sort(…,mode)：用 mode 选择排序方式，“ascend”为升序，“descend”为降序。

【例 8-25】 对随机产生的矩阵 A 分别进行降序和升序排序。

在命令窗口中输入如下语句。

```
A= unifrnd(1,2,4,5)
Y1=sort(A, 'descend')
Y2=sort(A, 'ascend')
```

命令窗口中的输出结果如下所示。

```
A =
    1.7943    1.6020    1.7482    1.9133    1.9961
    1.3112    1.2630    1.4505    1.1524    1.0782
    1.5285    1.6541    1.0838    1.8258    1.4427
    1.1656    1.6892    1.2290    1.5383    1.1067
Y1 =
    1.7943    1.6892    1.7482    1.9133    1.9961
    1.5285    1.6541    1.4505    1.8258    1.4427
    1.3112    1.6020    1.2290    1.5383    1.1067
```

```
    1.1656    1.2630    1.0838    1.1524    1.0782
Y2 =
    1.1656    1.2630    1.0838    1.1524    1.0782
    1.3112    1.6020    1.2290    1.5383    1.1067
    1.5285    1.6541    1.4505    1.8258    1.4427
    1.7943    1.6892    1.7482    1.9133    1.9961
```

8.3 数据插值

插值是图像处理和信号处理等领域常用的方法。在已知的数据之间寻找估计值的过程即为插值。MATLAB 2010 中提供了大量用于获取数据时间复杂度、空间复杂度及平滑度等的插值函数，这些函数保存在 MATLAB 2010 安装目录下的 ployfun 工具箱中，如表 8-6 所示。

表 8-6 插值函数

函 数 名	功 能 描 述
pchip	分段三次厄米多项式插值
interp1	一维插值
interp1q	一维快速插值
interpft	一维快速傅里叶插值
interp2	二维插值
interp3	三维插值
interpn	N 维插值
griddata	栅格数据插值
griddata3	三维栅格数据插值
griddatan	N 维栅格数据插值
spline	三次样条插值
ppval	分段多项式求值

8.3.1 一维插值

对一维函数进行插值即为一维插值。如图 8-7 所示，图中实心点表示已知数据点，空心点表示未知数据点，需要通过插值过程对横坐标 x 所对应的 y 值进行估计。在 MATLAB 2010 中存在两种类型的插值，即基于多项式的插值和基于傅里叶的插值。

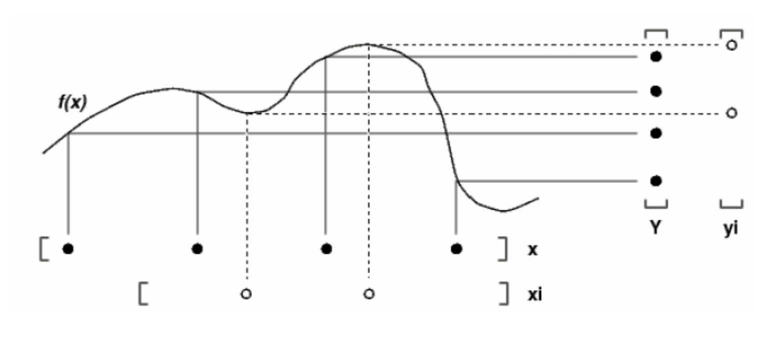


图 8-7 一维插值示意图

一维多项式插值可以通过函数 interp1()来实现，interp1()的格式如下。

- **yi=interp1(x,y,xi,method)**　其中 x 必须是矩阵，y 可以是向量也可以是矩阵。若 *y* 是向量，则必须与 x 长度相同，此时 xi 可以是标量、向量或任意维矩阵，yi 与 xi 大小相同；若 y 是矩阵，则其大小必须是[n,d1,d2,...,dk]，n 为向量 *x* 的长度，函数对 d1*d2*...*dk 组 y 值都进行插值。
- **yi=interp1(y,xi)**　默认 x 为 1:n，其中，n 为向量 y 的长度。
- **yi=interp1(x,y,xi,method)**　输入变量 method 用于指定插值的方法。
- **yi=interp1(x,y,xi,method,'extrap')**　对超出数据范围的插值数据指定外推方法。
- **yi=interp1(x,y,xi,method,extrapval)**　对超出数据范围的插值数据返回 extrapval 值，一般设为 NaN 或者 0。
- **yi=interp1(x,y,xi,'pp')**　返回值 pp 为数据 y 的分段多项式形式。method 指定产生分段多项式 pp 的方法，它可以是插值方法中除了的任何方法。

一维插值可以指定的方法如下。

（1）最邻近插值（method 取值为“neaest”），这种插值方法在已知数据的最邻近点设置插值点，对插值点的数进行四舍五入，对超出范围的点将返回 NaN。最邻近插值是最快的插值方法，但在数据平滑方面效果最差，得到的数据是不连续的。

（2）线性插值（method 取值为“linear”），该方法是未指定插值方法时所采用的方法，该方法直接连接相邻的两点，对超出范围的点将返回 NaN。比最邻近插值占用更多的内存，执行速度也稍慢，但在数据平滑方面优于最邻近插值，且线性插值的数据变化是连续的。

（3）三次样条插值（method 取值为“spline”），该方法采用三次样条函数来获得插值点。在已知点为端点的情况下，插值函数至少具有相同的一阶和二阶导数。处理速度最慢，占用内存小于分段三次厄米多项式插值，可以产生最光滑的结果，但在输入数据分布不均匀或数据点间距过近时将产生错误，样条插值是非常有用的插值方法。

（4）分段三次厄米多项式插值（method 取值为“pchip”），该参数取值的特点可参考帮助文档。在处理速度和内存消耗方面比线性插值差，但插值得到的数据和一阶导数是连续的。

（5）三次多项式插值（method 取值为“cublic”），该参数取值的特点可参考帮助文档。在处理速度和内存消耗方面比线性插值），与分段三次厄米多项式插值相同。

（6）三次多项式插值拟合已知数据（method 取值为“v5cublic”），该参数取值的特点可参考帮助文档。在处理速度和内存消耗方面比线性插值），该方法使用三次多项式函数对已知数据进行拟合。

上述方法的相对优劣不仅适用于一维插值，同样适用于二维或更高维度的情况。当选择一个插值方法时，应充分考虑其执行速度、内存占用及数据平滑度等方面的优劣。

以不同插值方法进行一维插值，代码设置如下。

```
%interp1_example.m
%用不同插值方法对一维数据进行插值，并比较其不同
x = 0:1.2:10;
y = sin(x);
xi = 0:0.1:10;
```

```
yi_nearest = interp1(x,y,xi,'nearset');     %最近邻插值
yi_linear = interp1(x,y,xi);                %默认插值方法是线性插值
yi_spline = interp1(x,y,xi,'spline ');      %三次样条插值
yi_cubic = interp1(x,y,xi,'cubic');         %三次多项式插值
yi_v5cubic = interp1(x,y,xi,'v5cubic');     %MATLAB5 中使用的三次多项式插值
hold on;
subplot(2,3,1);
plot(x,y,'ro',xi,yi nearest,'b-');
title('最近邻插值');
subplot(2,3,2);
plot(x,y,'ro',xi,yi linear,'b-');
title('线性插值');
subplot(2,3,3);
plot(x,y,'ro',xi,yi spline,'b-');
title('三次样条插值');
subplot(2,3,4);
plot(x,y,'ro',xi,yi cubic,'b-');
title('三次多项式插值');
subplot(2,3,5);
plot(x,y,'ro',xi,yi v5cubic,'b-');
title('三次多项式插值');
```

上述语句运行结果如图 8-8 所示。

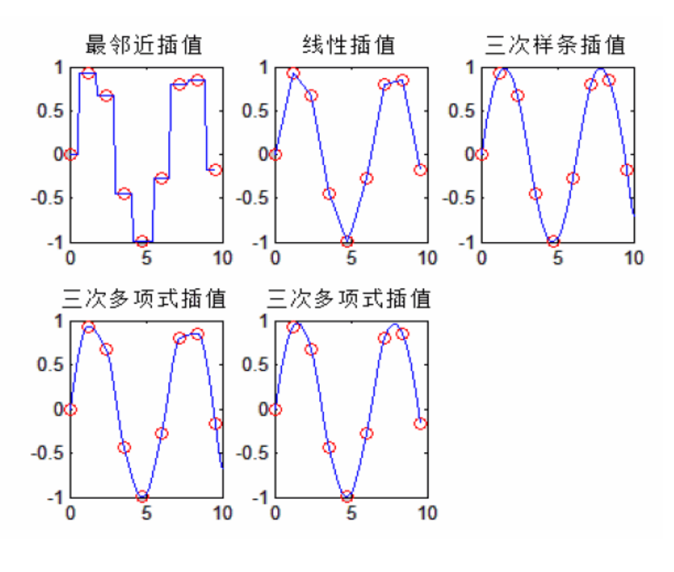


图 8-8　一维插值方法比较

一维快速傅里叶插值通过函数 interpft()来实现，该函数用傅里叶变换把输入数据变换到频域，然后用更多点的傅里叶逆变换，变换回时域，其结果是对数据进行增采样。函数 interpft()的调用格式如下。

- ❑ **y=interft(x,n)**　对 x 进行傅里叶变换，然后采用 n 点傅立叶逆变换变回到时域。如果 x 是一个向量，数据 x 的长度为 m，采样间隔为 dx，则数据 y 的采样间隔是 dx*m/n，注意 n 值必须大于 m；如果 x 是矩阵，函数操作在 x 的列上，返回结果与 x 具有相同的列数但其行数为 n。
- ❑ **y=interpft(x,n,dim)**　在 dim 指定的维度上进行操作。

例如，利用一维快速傅里叶插值实现数据增采样，代码设置如下。

```
%interpft example.m
%一维快速傅里叶插值实现数据增采样
x = 0:1.2:10;
y = sin(x);
n = 2*length(x);                %增采样 1 倍
yi = interpft(y,n);             %一维快速傅里叶插值
xi = 0:0.6:10.4;
hold on;
plot(x,y,'ro');                 %画图
plot(xi,yi,'b.-');
title('一维快速傅里叶插值');
legend('原始数据','插值结果');
```

由上述程序可生成图 8-9 所示结果。

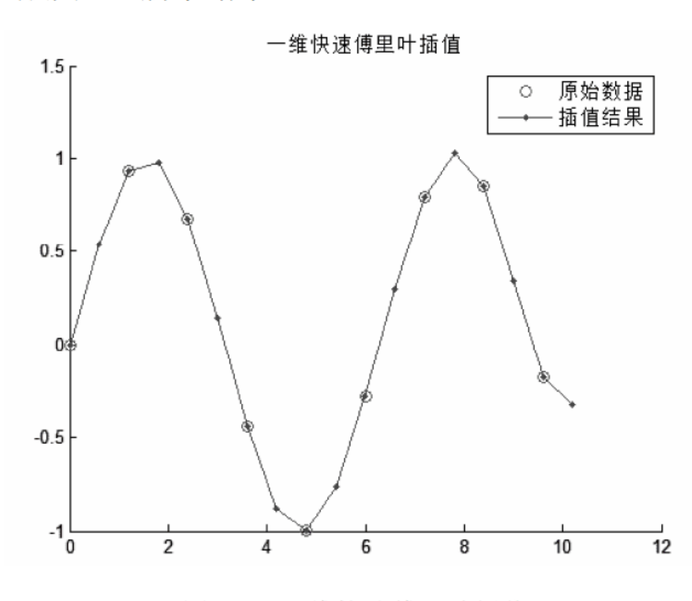


图 8-9　一维快速傅里叶插值

8.3.2　二维插值

二维插值主要应用于图像处理和数据可视化方面，其基本原理与一维插值一样，不同的是它是对两个变量的函数进行插值。MATLAB 中二维插值的函数为 interp2()，其调用格式如下。

- **zi=interp2(x,y,z,xi,yi)**　原始数据 x、y、z 决定插值函数，返回值 zi 是(xi,yi)在函数 f(x,y)上的值。
- **zi=interp2(z,xi,yi)**　若 $z=m\times n$，则 x=1，y=1:m。
- **zi=interp2(z,ntimes)**　在两点之间递归地插值 ntimes 次。
- **zi=interp2(x,y,z,xi,yi,method)**　可采用的不同的插值方法 method。
- **zi=interp2(method,extrapval)**　当数据超过原始数据范围时，用输入 extrapval 指定一种外推方法。

二维插值可采用的方法如下。

（1）最近邻插值（method='nearest'），这种插值方法在已知数据的最近邻点设置插值点，对插值的数进行四舍五入，对超出范围的点将返回一个 NaN。

（2）双线性插值（method='linear'），该方法是未指定插值方法时 MATLAB 默认采用的方法，插值点值取决于最近邻的四个点的值。

（3）三次样条插值（method='spline'），该方法采用三次样条函数来获得插值数据。

（4）双三次多项式插值（method='cublic'）。

下面通过实例比较各种二维插值方法的不同，具体如下。

```
%interp2 example2
%采用二次插值对三维高斯型分布函数进行插值
[x,y] = meshgrid(-3:0.8:3);                    %原始数据
z = peaks(x,y);
[xi,yi] = meshgrid(-3:0.25:3);                 %插值点
zi nearest = interp2(x,y,z,xi,yi,'nearest');   %最近邻插值
zi linear = interp2(x,y,z,xi,yi);              %默认插值方法是线性插值
zi spline = interp2(x,y,z,xi,yi,'spline');     %三次样条插值
zi cubic = interp2(x,y,z,xi,yi,'cubic');       %三次多项式插值
hold on;
subplot(2,3,1);
surf(x,y,z);
title('原始数据');
subplot(2,3,2);
surf(xi,yi,zi nearest);
title('最近邻插值');
subplot(2,3,3);
surf(xi,yi,zi linear);
title('线性插值');
subplot(2,3,4);
surf(xi,yi,zi spline);
title('三次样条插值');
subplot(2,3,5);
surf(xi,yi,zi cubic);
title('三次多项式插值');
figure;                                        %新开绘图窗口
subplot(2,2,1);                                %插值结果等高线
contour(xi,yi,zi nearest);
title('最近邻插值');
subplot(2,2,2);
contour(xi,yi,zi linear);
title('线性插值');
subplot(2,2,3);
contour(xi,yi,zi spline);
title('三次样条插值');
subplot(2,2,4);
contour(xi,yi,zi cubic);
title('三次多项式插值');
```

效果图分别如图 8-10、图 8-11 所示，图 8-11 中的插值等高线可用于比较插值后数据

的平滑性。

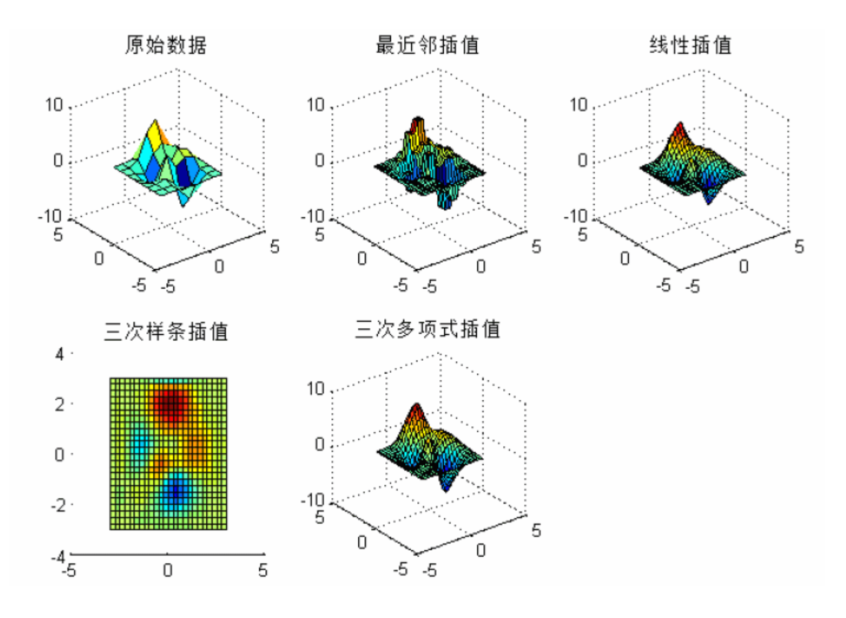


图 8-10　插值结果示意图

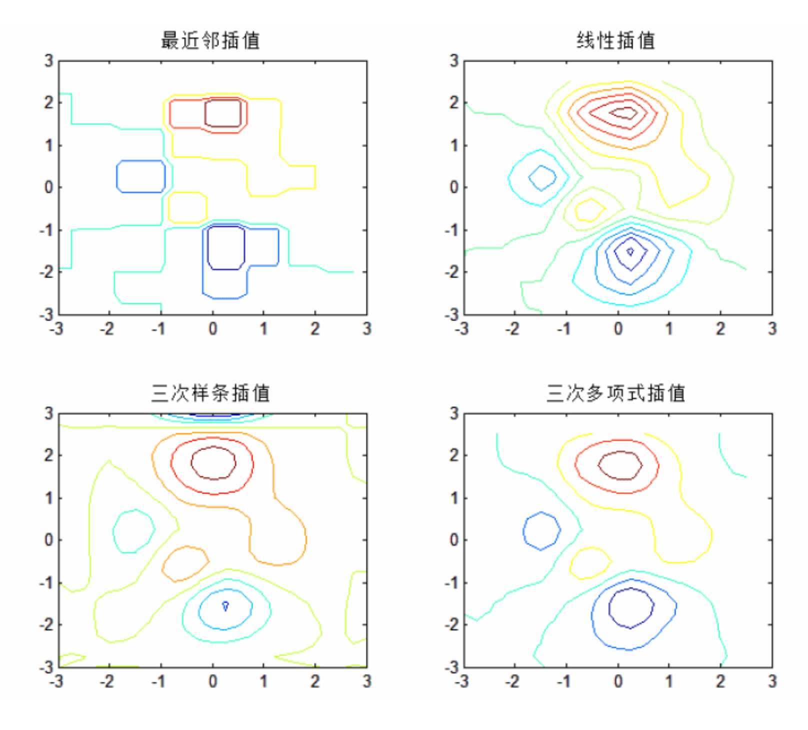


图 8-11　插值结果等高线示意图

8.3.3　三维插值

三维插值的思想与一维、二维插值基本相同，区别在于它是对三维函数的插值。MATLAB 提供了函数 interp3 实现三维插值，具体方法如下。

- ❑ **vi=interp3(x,y,z,v,xi,yi,zi)**　返回值 vi 是对于自变量(xi,yi,zi)的插值，需要说明的是，xi、yi、zi、vi 应为同维。

- vi=interp3(x,y,z,v,xi,yi,zi,method) method 用于指定插值方法，其取值及意义与二维插值的 method 方法相同。
- vi=interp3(x,y,z,v,xi,yi,zi,method,extrapval) 当数据超过原始数据范围时，输入 extrapval 指定一种外推方法。

下面通过实例对 flow(n)函数以不同方法实现三维插值，如 interp3_example 所示。

```
%interp3 example
[x,y,z,v] = flow(30);
[xi,yi,zi] = meshgrid(1:2:5, [0 1], [1 2]);
vi1= interp3(x,y,z,v,xi,yi,zi, 'nearest');
vi2= interp3(x,y,z,v,xi,yi,zi);
vi3= interp3(x,y,z,v,xi,yi,zi, 'spline');
vi4= interp3(x,y,z,v,xi,yi,zi, 'cubic');
figure
slice(x,y,z,v,2.5,[0.3 0.5],[1 1.5 2]);
title('原始数据');
figure
hold on;
subplot(2,2,1);
slice(xi,yi,zi,vi1, 2.5,[0.3 0.5],[1 1.5 2]);
title('最近邻插值');
subplot(2,2,2);
slice(xi,yi,zi,vi2, 2.5,[0.3 0.5],[1 1.5 2]);
title('线性插值');
subplot(2,2,3);
slice(xi,yi,zi,vi3, 2.5,[0.3 0.5],[1 1.5 2]);
title('三次样条插值');
subplot(2,2,4);
slice(xi,yi,zi,vi4, 2.5,[0.3 0.5],[1 1.5 2]);
title('三次多项式插值');
colormap hsv
```

图形窗口中显示三维插值的结果，如图 8-12、图 8-13 所示。

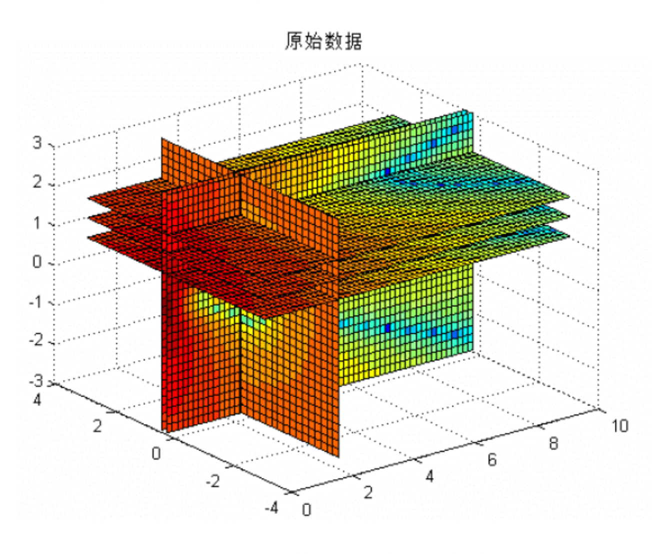


图 8-12　函数 flow 的原始数据

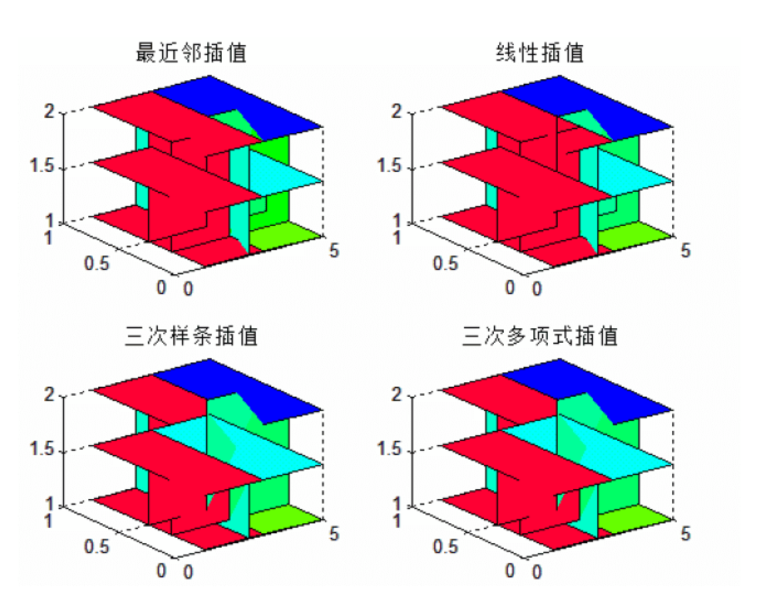


图 8-13　函数 flow 的三维差值结果

8.4　数值积分

MATLAB 2010 提供的数值积分函数及其功能描述如表 8-7 所示。

表 8-7　数值积分函数

参 数 名	功 能 描 述
quad	一元函数的数值积分，采用自适应 Simpson 方法
quadl	一元函数的数值积分，采用自适应 Lobatto 方法
quadv	一元函数的矢量数值积分
dblquad	二重积分
triplequad	三重积分

8.4.1　一元函数积分

MATLAB 2010 提供了两个函数 quad()和 quadl()计算一元函数的积分。函数 quad()采用低阶的自适应递归 Simpson 方法，函数 quadl()则采用高阶的自适应 Lobatto 方法。函数 quad()函数的调用格式如下。

- **q=quad(fun,a,b)**　计算函数 fun 在[a b]区间内的定积分。fun 为函数句柄，a 和 b 都是标量，它们分别是积分区间的下限和上限。
- **q=quad(fun,a,b,tol)**　以绝对误差容限 tol 计算函数 fun 在[a b]区间内的定积分，取代 MATLAB 中默认的绝对误差容限值。
- **q=quad(fun,a,b,tol,trace)**　当 trace 不为零时，显示计算定积分的迭代过程中向量[fcnt a b-a Q]。
- **q=quad(fun,a,b,...)**　多返回一个输出变量 fcnt，表示计算定积分过程中计算函数值的次数。

例如，求归一化高斯函数在区间[-1 1]上的定积分，代码设置如下。

```
%quad exam.m
%求归一化高斯函数在区间[-1 1]上的定积分，并求得到积分过程的中间节点
y=@(x)1/sqrt(pi)*exp(-x.^2);                    %归一化高斯函数
quad(y,-1,1,2e-6,1)                            %求定积分，并显示中间迭代过程
fplot(y,[-1 1],'b');                            %画出函数
hold on;
%跟踪数据(运行完上面程序后，可以在命令行复制这些数据)
trace = [    9       -1.0000000000    5.43160000e-001     0.1804679399;
            11       -1.0000000000    2.71580000e-001     0.0728222057;
            13       -0.7284200000    2.71580000e-001     0.1076454255;
            15       -0.4568400000    9.13680000e-001     0.4817487615;
            17       -0.4568400000    4.56840000e-001     0.2408826755;
            19       -0.4568400000    2.28420000e-001     0.1142172651;
            21       -0.2284200000    2.28420000e-001     0.1266655031;
            23        0.0000000000    4.56840000e-001     0.2408826755;
            25        0.0000000000    2.28420000e-001     0.1266655031;
            27        0.2284200000    2.28420000e-001     0.1142172651;
            29        0.4568400000    5.43160000e-001     0.1804679399;
            31        0.4568400000    2.71580000e-001     0.1076454255;
            33        0.7284200000    2.71580000e-001    0.0728222057];
x1 = trace(:,2);                               %积分过程的中间节点
y1 = y(x1);                                    %中间节点的函数值
plot(x1,y1,'ro');                              %画图
xlabel('x');
ylabel('y');
legend('高斯函数','求积分过程的中间节点');
```

由上述代码得到输出结果如下。

```
ans =
0.8427
```

积分过程如图 8-14 所示。

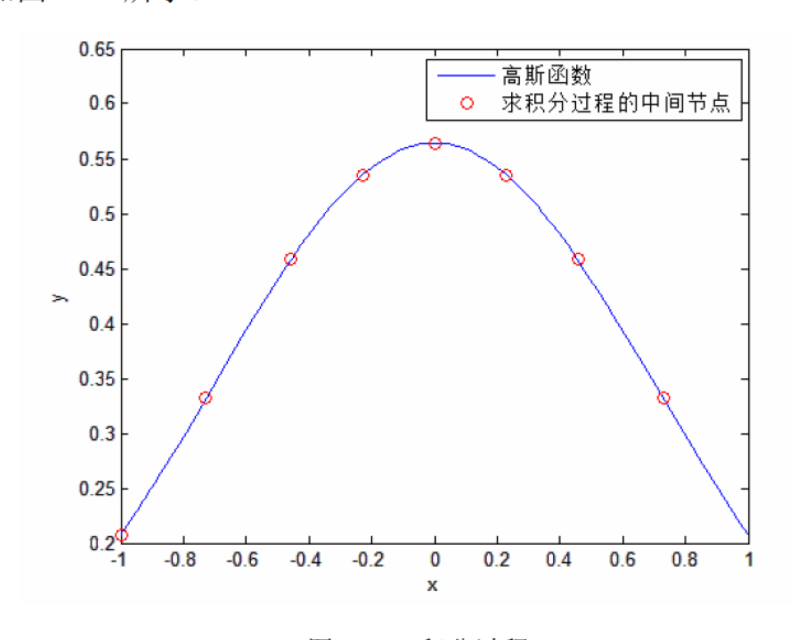


图 8-14　积分过程

函数 quadl()的调用格式与函数 quad()相同。

在一维积分的过程中，有可能出现三种警告信息。

- ❑ **Minimum step size reached** 已经达到了最小步长，这意味着积分的迭代区间比给定的定积分区间的截断误差值小。通常，这表明被积函数的可积性是奇异的。
- ❑ **Infinite or Not-a-Number function value encountered** 积分过程中出现除数为零或浮点数溢出。

8.4.2 矢量积分

矢量积分相当于多个一元定积分，如求 $\int_{-1}^{1}\frac{1}{\sqrt{2\pi * n}}\exp(-\frac{x^2}{2n^2})\mathrm{d}x, n=1,2,3,4,5$，就可以用矢量积分，具体代码如下。

```
%quadv exam.m
%求矢量积分
y=@(x,n)1./(sqrt(2*pi).*(1:n)).*exp(-x.^2./(2*(1:n).^2));  %归一化高斯函数
quadv(@(x)y(x,5),-1,1)
```

由上述语句得到的输出结果如下：

```
ans =
    0.6827    0.3829    0.2611    0.1974    0.1585
```

矢量积分的结果是一个向量，其每一元素的值为一个一元函数定积分的值。

8.4.3 二元函数积分

二元函数的积分形式如 $Q=\int_{y\min}^{y\max}\int_{x\min}^{x\max} f(x,y)\mathrm{d}x\mathrm{d}y$ 所示。

MATLAB 2010 中使用函数 dblquad()计算二重积分，该函数先计算内层积分值，然后利用内层积分的中间结果计算二重积分。根据 dxdy 的顺序，称 x 为内积分变量，y 为外积分变量。函数 dblquad()的基本格式如下。

- ❑ **q=dblquad(fun,xmin,xmax,ymin,ymax)** 计算二元函数 fun 在矩形区域[xmin,xmax,ymin,ymax]上二重积分。fun 为函数句柄，xmin、ymin、xmax 和 ymax 都是标量，它们分别是积分区间的下限和上限；
- ❑ **q=dblquad(fun,xmin,xmax,ymin,ymax,tol)** 其中，tol 参数用于指定绝对计算精度；
- ❑ **q=dblquad(fun,xmin,xmax,ymin,ymax,tol,method)** 用 method 方法指定计算一维积分时采用的函数。MATALB 默认采用函数 quad()计算一维积分。用户还可以编写自己的一维积分函数，以用于计算二维积分，该函数的调用格式须与 quad()一致。

例如，计算二维高斯函数在矩形区间[-1 1 -1 1]上的二重积分，代码设置如下。

```
%dblquad exam.m
%计算二维高斯函数在矩形区间[-1 1 -1 1]上的二重积分
f=@(x,y)1/sqrt(pi)*exp(-x.^2)*1/sqrt(pi)*exp(-y.^2);    %归一化高斯函数
dblquad(f,-1,1,-1,1,1e-6,@quadl)
```

由上述语句得到输出结果如下。

```
ans =
     0.7101
```

函数 dblquad()处理的都是矩形积分区域。若要计算非矩形的积分区间的二重积分，可以用一个大的矩形积分区域包含非矩形的积分区间，然后在非矩形的积分区间之外的区域上把二元函数的值取零。

例如，计算二维高斯函数在圆形区域$\sqrt{x^2+y^2}<1$上的二重积分，代码设置如下。

```
%dblquad exam2.m
%计算二维高斯函数在圆形区域 sqrt(x.^2+y.^2)<1 上的二重积分
%在圆形区域外填充 0 的归一化高斯函数
f=@(x,y)(1/sqrt(pi)*exp(-x.^2)*1/sqrt(pi)*exp(-y.^2)).*(sqrt(x.^2+y.^2)<=1);
dblquad(f,-2,2,-2,2,1e-6,@quadl)
```

由上述语句得到的输出结果如下。

```
ans =
     0.6321
```

8.4.4 三元函数积分

三重积分的形式如下，$Q=\int_{z\min}^{z\max}\int_{y\min}^{y\max}\int_{x\min}^{x\max}f(x,y,z)\mathrm{d}x\mathrm{d}y\mathrm{d}z$

三重积分可以用函数 triplequad()来实现，其用法与二重积分类似。

8.5 最优化问题求解

最优化问题是工程中经常遇到的问题，下面从无约束非线性极小化、有约束极小化、二次规划和线性规划、线性最小二乘、非线性最小二乘和多目标寻优方法等方面进行介绍。

8.5.1 无约束非线性极小化

在 MATLAB 中无约束非线性极小化问题的处理，使用的是 fminsearch 函数，具体方法如下。

- ❑ **x=fminsearch(fun,x0)**　fun 代表目标函数，x0 为初值，它可以是标量、向量或矩阵。
- ❑ **x=fminsearch(fun,x0,options)**　options 为进行优化的各种属性，由参数 optimset 函数设置。
- ❑ **x=fminsearch(...)**　x 代表极小值点，fval 代表最小值。

此外，还有一个 fminunc 函数，也是解决无约束非线性极小化问题的函数，其用法与 fminsearch 函数类似，该函数求的是局部解。

【例 8-26】 求解正弦函数在 x0 = 3 附近的极小值点，在命令窗口中输入如下语句。

```
clear
clc
X = fminsearch(@sin,3)
```

在命令窗口中的输出结果如下。

```
X =
   4.1724
```

8.5.2 有约束极小化

有约束条件的极小化问题相比于无约束条件极小化情况复杂很多，种类也比较繁多，这里只简单介绍函数 fmincon 的使用。

fmincon 函数用于解决如下约束条件的极小化问题：

```
min F(X)  subject to: A*X <= B, Aeq*X  = Beq  (线性约束)
                       C(X)<= 0, Ceq(X)  = 0  (非线性约束)
                       LB <= X <= UB         (边界)
```

fmincon 函数的具体用法如下。

- **x=fmincon(fun,x0,A,B)** fun 代表目标函数，x0 为初值，它可以是标量、向量或矩阵，线性约束条件为 A*B <= B 时，找到目标函数的极小值点。
- **x=fmincon(fun,x0,A,B,Aeq,Beq)** 线性约束条件为 A*X <= B 和 Aeq*X = Beq，无不等式存在时设 A=[]，B=[]。
- **x=fmincon(fun,x0,A,B,Aeq,Beq,LB,UB,nonlcon)** 计算非线性有约束条件(C(x) <= 0,Ceq(x) = 0)下 fun 函数的极小值点，如果没有边界则设定 LB = [],UB = []。

【**例 8-27**】求解约束条件下函数的极小值点。

```
clear
clc
X = fmincon(@(x)3*sin(x(1))+exp(x(2)),[1;1],[],[],[],[],[0 0])
```

命令窗口中的输出结果如下。

```
Active inequalities (to within options.TolCon = 1e-006):
  lower      upper     ineqlin   ineqnonlin
    1
    2
X =
     0
     0
```

8.5.3 二次规划和线性规划

1. 二次规划

用下式对二次规划的标准问题进行描述，其中 X、b、b_{eq}、lb 和 ub 都为列向量，A、A_{eq} 和 H 都为符合维数要求的矩阵。

$$\begin{cases} \min_X \left(\frac{1}{2}X'HX + f'X\right) \\ AX \leqslant b, A_{eq}X = b_{eq} \\ lb \leqslant X \leqslant ub \end{cases}$$

函数 quadprog 用来处理二次规划问题，该函数的具体用法如下。

- **x=quadprog(H,f,A,b)** 计算$\begin{cases} \min_X \left(\frac{1}{2}X'HX + f'X\right) \\ AX \leqslant b \end{cases}$线性规划的最优解。
- **x=quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)** 由指定设置 options 和初值 x0 开始计算$\begin{cases} z\min_X \left(\frac{1}{2}X'HX + f'X\right) \\ AX \leqslant b, A_{eq}X = b_{eq} \\ lb \leqslant X \leqslant ub \end{cases}$线性规划的最优解，其中 options 可以由函数 optimset 设定。
- **[x,fval]=(H,f,A,b,Aeq,beq,lb,ub,x0,options)** 附加返回最小值。

【例 8-28】 计算$\begin{cases} \min\left(\frac{1}{3}(x_1x_2)\begin{pmatrix} 1 & -1 \\ -1 & 2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} - 3x_1 - 5x_2\right) \\ x_1 + 3x_2 \leqslant 2 \\ -x_1 + x_2 \leqslant 8 \\ x_1, x_2 \geqslant 0 \end{cases}$的线性规划。

在命令窗口中输入如下语句。

```
H = [1 -1; -1 2];
f = [-3; -5];
A = [1 3;-1 2];
b = [2; 8];
[x,fval] = quadprog(H,f,A,b,[],[],[0;0])
```

命令窗口中的输出结果如下。

```
Warning: Large-scale method does not currently solve this problem
formulation,
using medium-scale method instead.
> In quadprog at 263
Optimization terminated.
x =
    1.2941
    0.2353
fval =
    -4.4706
```

2. 线性规划

用下式对线性规划问题进行数学描述，其中 X、f、b、b_{eq}、lb 和 ub 都为列向量，A 和 A_{eq} 都为符合维数要求的矩阵。

$$\begin{cases} \min_X f^T X \\ AX \leqslant b, A_{eq}X = b_{eq} \\ lb \leqslant X \leqslant ub \end{cases}$$

函数 linprog 用来处理线性规划问题，下面详细介绍该函数的具体用法。

- **x=linprog(f,A,b)**　计算 $\begin{cases} \min_X f^T X \\ AX \leqslant b \end{cases}$ 线性规划的最优解。
- **x=linprog(f,A,b,Aeq,beq)**　计算 $\begin{cases} \min_X f^T X \\ AX \leqslant b, A_{eq}X = b_{eq} \end{cases}$ 线性规划的最优解。
- **x=linprog(f,[],[],Aeq,beq)**　计算 $\begin{cases} \min_X f^T X \\ AX = b_{eq} \end{cases}$ 线性规划的最优解。
- **x=linprog(f,A,b,Aeq,beq,lb,ub)**　计算 $\begin{cases} \min_X f^T X \\ AX \leqslant b, A_{eq}X = b_{eq} \\ lb \leqslant X \leqslant ub \end{cases}$ 线性规划的最优解。
- **x=linprog(f,A,b,Aeq,beq,lb,ub,x0)**　由初值 x0 计算 $\begin{cases} \min_X f^T X \\ AX \leqslant b, A_{eq}X = b_{eq} \\ lb \leqslant X \leqslant ub \end{cases}$ 线性规划的最优解。
- **x=linprog(f,A,b,Aeq,beq,lb,ub,x0,options)**　由指定设置 options 及初值 x0 开始计算 $\begin{cases} \min_X f^T X \\ AX \leqslant b, A_{eq}X = b_{eq} \\ lb \leqslant X \leqslant ub \end{cases}$ 线性规划的最优解，其中 options 由函数 optimset 设置，参见 MATLAB 帮助。
- **[x,fval]=linprog(f,A,b,Aeq,beq,lb,ub,x0)**　附加返回 $f^T X$。

【例 8-29】 计算 $\begin{cases} \min(x_1 + 3x_2 - 5x_3) \\ x_1 - x_2 + x_3 \leqslant 8 \\ 3x_1 + x_3 \leqslant 4 \\ x_1, x_2, x_3 \geqslant 0 \end{cases}$ 的线性规划。

在 MATLAB 的命令窗口中输入如下语句。

```
f = [1;3;-5];
A = [1 -1 1;3 0 1];
b = [8;4];
x = linprog(f,A,b,[],[],[0;0;0],[inf;inf;inf])
```

则命令窗口中显示输出结果如下。

```
Optimization terminated.
x =
   0.0000
```

```
    0.0000
    4.0000
```

值得注意的是，有时由于约束条件不够，往往无法找到最优解。

此外，MATLAB 还提供了函数 bintprog 进行二进制整数规划，但未提供进行整数规划的函数。



8.5.4　线性最小二乘

利用左除（\）可以求解线性最小二乘问题。若 A 为 m×n 矩阵(m≠n)，且 b 为具有 m 个元素的列向量或具有多个此类问题向量的矩阵，则 X = A \ B 为等式 AX = b 的最小二乘意义上的解。

1. 非负线性最小二乘

非负线性最小二乘问题的数学描述如下所示。

$$\min_x \frac{1}{2}\| Cx - d \|_2^2 \qquad x \geqslant 0$$

其中，矩阵 C 和向量 d 为目标函数的系数。

在 MATLAB 中用 lsqnonneg 函数求线性问题的非负最小二乘解，具体用法如下所示。

- ❑ **x=lsqnonneg(C,d)**　返回向量 x，使得范数(C*x-d)最小化，约束条件为 x≥0。C 和 d 必须为实数。
- ❑ **x=lsqnonneg(C,d,x0)**　若所有的 x0≥0，则使用 x0 为初值，否则使用默认值。默认的初值为原点(当 x0==[]或只提供两个输入变量时也使用默认值)。
- ❑ **x=lsqnonneg(C,d,x0,options)**　用 options 结构指定的优化参数进行最小化。
- ❑ **x=lsqnonneg(...)**　返回残差的平方范数值 norm = (C*x − d)^2 。

【例 8-30】　通过问题比较 lsqnonneg 函数解与无约束最小二乘解的不同。

在命令窗口中输入如下语句。

```
C = [0.0372 0.2869;0.6861 0.7071;0.6233 0.6245;0.6344 0.6170];
d = [0.8587;0.1781;0.0747;0.8405];
x1 = lsqnonneg(C,d)
x2 = C\d
```

则命令窗口显示输出结果如下。

```
x1 =
     0
     0.6929
x2 =
     -2.5627
      3.1108
```

由结果可知二者的解不一样，肺腑最小二乘解没有负值。

2. 有约束线性最小二乘

有约束线性最小二乘问题的数学描述如下所示。

$$\begin{cases} \min_x \frac{1}{2}\|Cx-d\|_2^2 \\ A\cdot X \leqslant b \\ Aeq\cdot x = beq \\ lb \leqslant x \leqslant ub \end{cases}$$

其中，C、A 和 Aeq 为矩阵，d、b、beq、lb、ub 和 x 为向量。在 MATLAB 中用 lsqlin 函数求有约束线性最小二乘解，具体如下。

- ❑ **x=lsqlin(C,d,A,b)** 求解最小二乘意义上的线性系统 C*x = d，约束条件为 A*X ⩽ b，其中 C 为 m*n 的矩阵。
- ❑ **x=lsqlin(C,d,A,b,Aeq,beq)** 加上等式约束 Aeq*x = beq 以后求解上面的问题，如果没有不等式存在，则令 A=[],b=[]。
- ❑ **x=lsqlin(C,d,A,b,lb,ub)** 为 x 定义一系列下界 lb 和上界 ub 使得总有 lb ⩽ x ⩽ ub 。
- ❑ **[x,resnorm,residual,exitflag,output]=lsqlin(...)** 返回与优化信息有关的结构输出参数 output。

【例 8-31】 求解下列问题的最小二乘解。

$$C\cdot x = d$$
$$A\cdot x \leqslant b$$
$$lb \leqslant x \leqslant ub$$

在命令窗口中输入如下代码。

```
C =
[0.9501 0.7620 0.6153 0.4057;
0.2311 0.4564 0.7919 0.9354;
0.6068 0.0185 0.9218 0.9169;
0.4859 0.8214 0.7382 0.4102;
0.8912 0.4447 0.1762 0.8936];
d = [0.0578;0.3528;0.8131;0.0098;0.1388];
A =
    [0.2027 0.2721 0.7467 0.4659;
    0.1987 0.1988 0.4450 0.4186;
    0.6037 0.0152 0.9318 0.8462];
b = [0.5251;0.2026;0.6721];
lb = -0.1*ones(4,1);
ub = 2*ones(4,1);
x = lsqlin(C,d,A,b)
```

命令窗口中的输出结果如下。

```
Warning: Large-scale method can handle bound constraints only;
using medium-scale method instead.
> In lsqlin at 289
Optimization terminated.
x =
     0.1299
    -0.5757
     0.4251
     0.2438
```

8.5.5 非线性最小二乘

非线性最小二乘问题的数学描述如下。

$$\min_X f(x) = f_1(x)^2 + f_2(x)^2 + \cdots + f_m(x)^2 + L$$

式中，L 为常数。

在 MATLAB 中，用 lsqnonlin 函数求解非线性最小二乘问题，具体用法如下。

- **x=lsqnonlin(fun,x0)** 初值为 $X0$，求 fun 函数的最小平方和。fun 函数将返回一个数值向量但不是值的平方。
- **x=lsqnonlin(fun,x0,lb,ub)** 定义一系列的下界 lb 和上界 ub，使得总有。
- **x=lsqnonlin(fun,x0,lb,ub,options)** 用 options 结构指定的优化参数进行最小化。
- **x=lsqnonlin(fun,x0,lb,ub,options,P1,P2,...)** 将问题参数 P1、P2 等直接传递给 fun 函数，将空矩阵传递给 options 参数作为其默认值。
- **[x,resnorm]=lsqnonlin(...)** 返回 x 处残差的平方范数值 sum(fun(x).^2)。

【例 8-32】 求解 x，使得下式最小化。

$$\sum_{k=1}^{10}(2+2k-e^{kx_1}-e^{kx_2})^2\text{，初值为 } x=[0.3\ 0.4]$$

由于 lsqnonlin 函数假设用户提供的平方和不是显式表达的，因此，lsqnonlin 函数中的函数应变换为向量值函数，如下所示。

$$F_x(x) = 2+2k-e^{kx_1}-e^{kx_2}, k=1,2,3,\ldots,10$$

编辑 M 文件并保存为 myfun1.m 文件，程序代码如下所示。

```
function F=myfun1(x)
k=1:10;
F=2+2*k-exp(k*x(1))-exp(k*x(2));
```

其次在命令窗口中输入如下语句。

```
x0=[0.3 0.4]
[x,resnorm]=lsqnonlin(@myfun1,x0)
```

则命令窗口中的输出结果如下。

```
x0 =
    0.3000    0.4000
Optimization terminated: norm of the current step is less
 than OPTIONS.TolX.
x =
    0.2578    0.2578
resnorm =
  124.3622
```

8.5.6 多目标寻优方法

多目标寻优方法与前面的只有一个目标函数的单目标优化方法不同。在许多实际工程问题中，我们往往希望多个指标均达到最优值，即使得多个目标函数均达到最优值，此类问题被称之为多目标寻优方法，其数学描述如下。

$$\begin{cases} \min\limits_{x\in R^n} F(x) \\ G_i(x)=0, i=1,\ldots,m_e \\ G_i(x)\leqslant 0, i=m_e+1,\ldots,m \\ x_i\leqslant x\leqslant x_u \end{cases}$$

其中，$F(x)$为目标函数。

由于多目标寻优问题往往没有唯一解，因此必须引进非劣解的概念。

若 $x^*\in\Omega$，且对于 x^*不存在 Δx，使得 $(x^*+\Delta x)\in\Omega$ 与

$$F_i(x^*+\Delta x)\leqslant F_i(x^*), i=1,\ldots,m$$

$$f_j(x^*+\Delta x)F_j(x^*),\exists j$$

不能同时成立，那么定义 x^*为多目标寻优问题的非劣解。

1. 多目标寻优揭发

多目标寻优有多种解法，常用的形式如下。

1）权和法

该方法将多目标向量问题转化为所有目标的加权求和的标量问题，数学描述如下。

$$\min_{x\in\Omega} f(x)=\sum_{i=1}^{m}\omega_i\cdot F_p(x)^2$$

其中 ω_i 为加权因子，它的选取方法很多，如加权因子分解法和容限法等。

2）约束法

约束法对目标函数向量中的主要目标 F_p 进行最小化，将其他目标不等式约束的形式写出。

$$\min_{x\in\Omega}\cdot F_p(x)^2$$

$$sub.F_i(x)\leqslant\varepsilon_i, i=1,\ldots,m; i\neq p$$

3）目标达到法

目标函数为 $F(x)=\{F_1(x),F_2(x),\ldots F_m(x)\}$，对应目标值为 $F^*=\{F_1^*,F_2^*,\ldots,F_m^*)$。允许目标函数有正负偏差，偏差的大小由加权系数 $W=(W_1,W_2,\ldots,W_m)$ 控制，于是目标达到问题就可以表述为标准的寻优问题。

$$\min_{x\in\Omega,\gamma\in R}\gamma$$

$$sub.F_i(x)-\omega_i\gamma\leqslant F_i^*, i=1,\ldots m$$

指定目标 $\{F_1^*,F_2^*\}$，定义目标点 P。权重向量定义从 P 到可行域空间 $\Lambda(\gamma)$ 的搜索方向。在寻优过程中，γ 的变化改变可行域的大小，约束边界变为唯一解点 (F_{1s},F_{2s})。

4）目标达到法的改进

目标达到法的一个好处是可以将多目标寻优问题转化为非线性规划问题。通过将目标达到问题变为最大最小化问题来获得更合适的目标函数：$\min\limits_{x\in R^n}\max\limits_{i}\{\Lambda_i\}$，其中，$\Lambda_i=\dfrac{F_i(x)-F_i^*}{W_i}, i=1,\ldots,m$。

2. 多目标寻优的有关函数

多目标达到寻优的数学描述如下所示。

$$\min_{x,\gamma} \gamma$$
$$F(x) - weight \cdot \gamma \leqslant goal$$
$$c(x) \leqslant 0$$
$$ceq(x) = 0$$
$$A \cdot x \leqslant b$$
$$Aeq \cdot x = beq$$
$$lb \leqslant x \leqslant ub$$

其中，*x*、*weight*、*goal*、*b*、*beq*、*lb*、*ub* 为向量，*A*、*Aeq* 为矩阵，*c*(*x*)、*ceq*(*x*)、*F*(*x*)为函数，返回向量。*c*(*x*)、*ceq*(*x*)、*F*(*x*)也可以是非线性函数。

在 MATLAB 中，利用 fgoalattain 函数来处理多目标寻优问题，具体用法如下所示。

- ❑ **x=fgolaattain(fun,x0,goal,weight)** 试图通过改变 x 来使目标函数 fun 达到 goal 指定的目标。初值为 x0，weight 参数指定权重。
- ❑ **x=fgolaattain(fun,x0,goal,weight,A,b)** 求解目标寻优问题，约束条件为线性不等式 $A*x \leqslant b$。
- ❑ **x=fgolaattain(fun,x0,goal,weight,A,b,Aeq,beq)** 求解目标寻优问题，除了 $\mathrm{A*x \leqslant b}$ 外，还有 $\mathrm{Aeq*x = beq}$，若无不等式存在，则设置 $\mathrm{A = [],b = []}$。
- ❑ **x=fgolaattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub)** 为设计变量 x 定义下界 lb 和上界 ub 集合，使得 $\mathrm{lb \leqslant x \leqslant ub}$。
- ❑ **x=fgolaattain(fun,x0,goal,weight,A,b,Aeq,beq,lb,ub,nonlcon)** nonlcon 是一个用户定义的函数 $\mathrm{function[c,ceq] = mycon(x)}$，根据状态向量 *x* 计算非线性约束 $\mathrm{c(x) \leqslant 0}$ 和非线性等式约束 $\mathrm{ceq(x) = 0}$，若不存在边界，则设置 $\mathrm{lb = []}$，$\mathrm{ub = []}$。

下面以实例说明多目标寻优问题的解决方法。

【例 8-33】 某玩具厂制作两种不同的涂料产品 A 和 B，已知生产 A 涂料 100 kg 需要 8 个工时，生产 B 涂料 100 kg 需要 10 个工时。限定每日的工时数为 40，并希望不需要临时工，也不需要工人加班生产。这两种涂料每 100kg 可获利 100 元。此外，有个顾客需求供应 B 涂料 600 kg，请问应如何制定生产计划，达到最优？

分析：假设制作 A 和 B 两种涂料的数量分别为 x_1、x_2（均以 100kg 计），为了使生产计划比较合理并用人尽量少，使利润最大化，并且 B 涂料的产量尽量多，由以上分析可建立如下所示的数字描述。

$$\min z_1 = 8x_1 + 10x_2$$
$$\min z_2 = 100x_1 + 100X_2$$
$$\min z_3 = x_2$$
$$8x_1 + 10x_2 \leqslant 40$$
$$x_2 \geqslant 6$$
$$x_1, x_2 \geqslant 0$$

编写目标函数的 M 文件，保存 goal.m，返回目标计算值。

```
function f=myfun(x)
f(1)=8*x(1)+10*x(2);
f(2)=100*x(1)-100*x(2);
f(3)=-x(2);
```

给定目标，权重按目标比例确定，给出初始值。

```
goal=[400 -800 -6];
weight=[400 -800 -6];
x0=[2 2];
```

给出约束条件的系数。

```
A=[8 10;0 -1];
b=[40 -6];
lb=zeros(2,1);
options=optimset('MaxFunEvals',5000);    %函数将最大次数设置为 5000 次
[x,fval,attainfactor,exitflag]=...
fgoalattain(@(x)goal,x0,goal,weight,A,b,[],[],lb,[],[],options);
```

命令窗口中的输出结果如下。

```
x =
   2.0452    1.9429
fval =
        400   -800    -6
attainfactor =
           -0.0681
exitflag =
           0
```

由结果可知，经过 5 000 次迭代以后，生产 A、B 涂料的数据量分别为 204.52kg 和 194.29kg。

8.6 本章小结

本章主要介绍经常用到的 MATLAB 科学计算问题的求解方法。分别介绍了包括线性方程与非线性方程及常微分方程的求解、数据统计处理、数据插值、数值积分和优化问题求解等方面的内容。对每一类计算问题，分别通过对实例的详细分析，进而加深读者对求解方法的理解。

8.7 习题

（1）计算一元函数 $f(x)=3x^2\sin x-4x$ 在[-1,10]区间的零点。

（2）计算积分 $\int_{-3}^{3}\frac{1}{n}\exp\left(-\frac{2x}{n^3}\right)dx, n=1,2,3,4$ 。

（3）计算 $\begin{cases}\min\left(\frac{1}{3}(x_1x_2)\begin{pmatrix}1 & -1\\ -1 & 2\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}-3x_1-5x_2\right)\\ x_1+4x_2\leqslant 3\\ -x_1+3x_2\leqslant 10\\ x_1,x_2\geqslant 0\end{cases}$ 的线性规划。

第 9 章　S-函数

S-函数，即 S-Function，是 system-function 的缩写。简单地说，S-函数就是用 MATLAB 所提供的模型不能完全满足用户，而提供给用户自己编写程序来满足自己要求模型的接口。

S-Function 可以使用 MATLAB、C、C++、Ada 或 Fortran 语言来编写。使用 MEX 实用工具，将 C、C++、Ada 和 Fortran 语言的 S-Function 编译成 MEX-文件，在需要时，它们可与其他 MEX-文件一起动态地连接到 MATLAB 中。

S-函数由一种特定的语法构成，用来描述并实现连续系统、离散系统及复合系统等动态系统；S-函数能够接受来自 Simulink 求解器的相关信息，并对求解器发出的命令作出适当的响应，这种交互作用非常类似于 Simulink 系统模块与求解器的交互作用。一个结构体系完整的 S-函数包含了描述动态系统所需的全部能力，所有其他的使用情况都是这个结构体系的特例。往往 S-函数模块是整个 Simulink 动态系统的核心。

S-函数作为与其他语言相结合的接口，可以使用这个语言所提供的强大能力。例如，MATLAB 语言编写的 S-函数可以充分利用 MATLAB 所提供的丰富资源，方便调用各种工具箱函数和图形函数；使用 C 语言编写的 S-函数可以实现对操作系统的访问，如实现与其他进程的通信和同步等。

9.1　基本概念

理解下列与 S-函数相关的一些基本概念对于用户理解 S-函数的概念与编写都是非常有益的；而且这些概念在其他的仿真语言中也是会经常遇到的。

1．仿真例程（Routines）

Simulink 在仿真的特定阶段调用对应的 S-函数功能模块（函数），来完成不同的任务，如初始化、计算输出、更新离散状态、计算导数和结束仿真等，这些功能模块（函数）称为仿真例程或回调函数（Callback Functions）。表 9-1 列出了 S-函数的例程函数和对应的仿真阶段。

表 9-1　函数例程

S-函数仿真例程	仿 真 阶 段
mdlInitialization	初始化
mdlGetTimeOfNextVarHit	计算下一个采样点
mdlOutput	计算输出
mdlUpdate	更新离散状态
mdlDerivatives	计算导数
S-函数仿真例程	仿真阶段
mdlTerminate	结束仿真

2．直接馈通（Direct feedthrough）

直接馈通意味着输出或可变采样时间与输入直接相关。在如下的两种情况下需要直接馈通。

（1）某一时刻的系统输出 y 中包含某一时刻的系统输入 u。

（2）系统是一个变采样时间系统（variable sample time system）且采样时间计算与输入 *u* 相关。

$$\text{time}=(\text{n}\times\text{sample_time_value})+\text{offset}+\text{time}$$

其中，n 表示第 n 个采样点。

Simulink 在每一个采样点上调用 mdlOutput 和 mdlUpdate 例程。对于连续时间系统采样时间和偏移量的值应该设置为零。采样时间还可以继承自驱动模块、目标模块或系统最小采样时间，这种情况下采样时间值应该设置为-1，或者 INHERITED_SAMPLE_TIME。

3．动态输入（Dynamically sized inputs）

S-函数支持动态可变维数的输入。S-函数的输入变量 u 的维数决定于驱动 S-函数模块的输入信号的维数。

9.2 工作原理

在对动态系统建模时，总是能够采用广义的状态空间形式对无论是线性系统还是非线性系统进行描述，这个描述包含两个方程，即状态方程与输出方程。

状态方程描述了状态变量的一阶导数与状态变量、输入量之间的关系。n 阶系统具有 n 个独立的状态变量，系统状态方程则是 n 个联立的一阶微分方程或差分方程。对于一个系统，由于所选择的状态变量不同，会导出不同的状态方程，因此状态方程的形式不是唯一的。输出方程描述了输出与状态变量、输入量之间的关系。输出量根据任务的需要确定。一个典型的线性系统的状态方程可以用矩阵的形式描述为：

$$\begin{cases} x' = Ax + Bu \\ y = Cx + Du \end{cases}$$

这里，$x=\begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$，$u=\begin{bmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{bmatrix}$ $y=\begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

$$A=\begin{bmatrix} a_{11} & \cdots & a_{1n} \\ \vdots & & \vdots \\ a_{n1} & & a_{nn} \end{bmatrix}_{n\times n} \qquad B=\begin{bmatrix} b_{11} & \cdots & b_{1r} \\ \vdots & & \vdots \\ b_{n1} & & b_{nr} \end{bmatrix}_{n\times r}$$

$$C=\begin{bmatrix} c_{11} & \cdots & c_{1n} \\ \vdots & & \vdots \\ c_{m1} & & c_{mn} \end{bmatrix}_{m\times n} \qquad D=\begin{bmatrix} d_{11} & \cdots & d_{1r} \\ \vdots & & \vdots \\ d_{m1} & & d_{mr} \end{bmatrix}_{m\times r}$$

其中 A、B、C、D 分别是状态矩阵、输入矩阵、输出矩阵、前馈矩阵。Simulink 框图

的大部分模块都具有一个输入向量 u、一个输出向量 y 和一个状态向量 x，如图 9-1 所示。

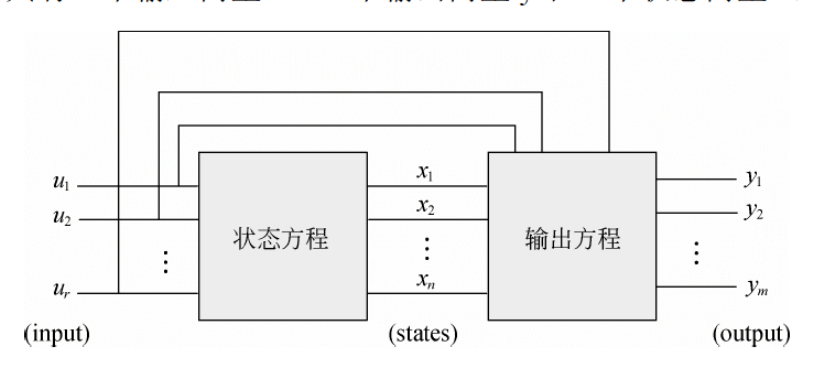


图 9-1　Simulink 模块

1．S-函数的例程函数

S-函数同样是一个 Simulink 模块。它的以下几个例程函数清楚地体现了状态空间所描述的特性。

（1）S-函数中的连续状态方程描述。状态向量的一阶导数是状态 **x**、输入 **u** 和时间 t 的函数。在 S-函数中，状态的一阶导数是在 mdlDerivatives 例程中计算的，并将结果返回供求解器积分。

（2）S-函数中的离散状态方程描述。下一步状态的值依赖于当前的状态输入 **u** 和时间 t。这是通过 mdlUpdate 例程完成的，并将结果返回供求解器在下一步时使用。

（3）S-函数中的输出方程描述。输出值是状态、输入和时间的函数。

2．Simulink 仿真的两个阶段

理解 S-函数首先要很好地了解 Simulink 的仿真过程。仿真包含两个主要阶段，第一个阶段是初始化，这时块的所有参数都已确定下来。初始化阶段完成了以下工作。

- ❑ 传递参数给 MATLAB 进行求值。
- ❑ 得到的数值作为实际的参数使用。
- ❑ 展开模型的层次，每个子系统被它们所包含的块替代。
- ❑ 检查信号的宽度和连接。
- ❑ 确定状态初值和采样时间。

仿真的第二个阶段是运行阶段，工作可以概括如下。

（1）计算输出。

（2）更新离散状态。

（3）计算连续状态，连续状态的计算过程如下。

① 每个块按照预先确定的顺序计算输出。

② 每个块使用当前时间、块的输入和状态计算其导数。

③ 导数返回给求解器，通过积分得到下一步状态的值。

（4）计算输出，过零可能被激活。

3．S-函数仿真的两个阶段

S-函数是 Simulink 的重要组成部分，其仿真过程包含在 Simulink 仿真过程之中，如图 9-2 所示，S-函数的仿真流程也包括初始化阶段和运行阶段两个阶段。图 9-2 中每个功能模

块都对应于一个仿真例程或回调函数。

（1）初始化：在仿真开始前，Simulink在这个阶段初始化 S-函数。

① 初始化结构体 SimStruct，它包含了S-函数的所有信息。

② 设置输入输出端口数。

③ 设置采样时间。

④ 分配存储空间。

（2）计算下一个采样时间点：只有在使用变步长求解器进行仿真时，才需要计算下一个采样时间点，即计算下一步的仿真步长。

（3）计算输出：计算所有输出端口的输出值。

（4）更新状态：此例程在每个步长处都要执行一次，可以在这个例程中添加每一个仿真步都需要更新的内容，如离散状态的更新。

（5）数值积分：用于连续状态的求解和非采样过零点。如果 S-函数存在连续状态，Simulink 就在 minor step time 内调用 mdlDdrivatives 和 mdlOutput 两个 S-函数例程。

图 9-2　S-函数仿真流程

9.3　Level-1 M 文件型

用 M 语言编写的 S-函数称为 M 文件的 S 函数，根据 API 版本不同，分为 Level-1 M 文件型 S-函数和 Level-2 M 文件型 S 函数。

本节将通过 Level-1 M 文件型 S-函数的概述、编写方法及大量实例介绍该类型的 S 函数，以满足用户不同的需求。

9.3.1　概述

在 MATLAB 命令窗口中输入命令 sfundemos，可以查看 S-函数示例，如图 9-3 所示，其中列出了多种方式书写的 S-函数的实例。

双击“M-files”，可以看到如图 9-4 所示的界面，其中列出了用两种方式书写的 M 文件型 S-函数的实例。

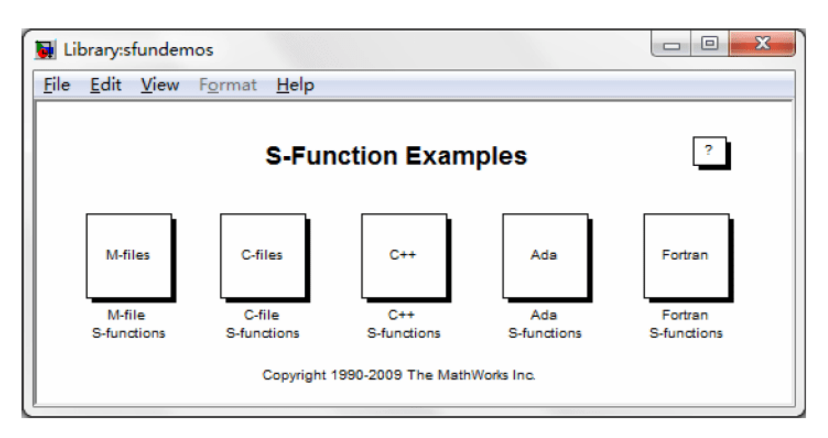


图 9-3 S-函数实例演示模块

继续双击“Level-1 M-files”，可以看到如图 9-5 所示的界面，其中列出了 Level-1 M 文件型 S-函数的大量实例。

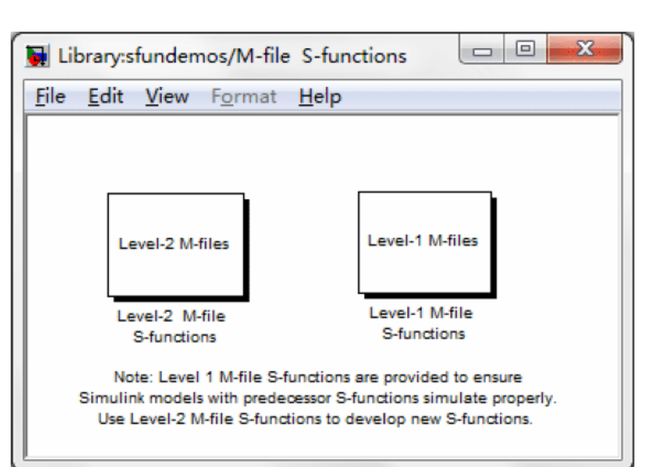


图 9-4 M 文件 S 函数实例演示模块

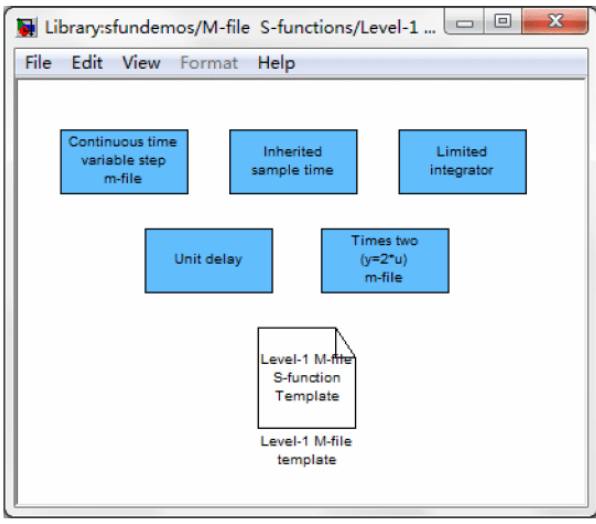


图 9-5 Level-1 M 文件型 S-函数实例演示模块

如图 9-5 所示包含一个 Level1 M-file S-function Template 示例，它提供了书写该类型 S-函数的模板，删除注释后的具体内容如下。

```
function [sys,x0,str,ts,simStateCompliance] = sfuntmpl(t,x,u,flag)
switch flag,
      case 0,
        [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes;
      case 1,
        sys=mdlDerivatives(t,x,u);
      case 2,
        sys=mdlUpdate(t,x,u);
      case 3,
        sys=mdlOutputs(t,x,u);
      case 4,
        sys=mdlGetTimeOfNextVarHit(t,x,u);
  case 9,
        sys=mdlTerminate(t,x,u);
      otherwise
        DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
     end
function [sys,x0,str,ts,simStateCompliance]=mdlInitializeSizes
    sizes = simsizes;
```

```
    sizes.NumContStates = 0;
    sizes.NumDiscStates = 0;
    sizes.NumOutputs = 0;
    sizes.NumInputs = 0;
    sizes.DirFeedthrough = 1;
    sizes.NumSampleTimes = 1;   % at least one sample time is needed

    sys = simsizes(sizes);

    x0 = [];
    str = [];
    ts = [0 0];
    simStateCompliance = 'UnknownSimState';
    function sys=mdlDerivatives(t,x,u)
    sys = [];
    function sys=mdlUpdate(t,x,u)
    sys = [];
    function sys=mdlOutputs(t,x,u)
    sys = [];
    function sys=mdlGetTimeOfNextVarHit(t,x,u)
    sampleTime = 1;    %  Example, set the next hit to be one second later.
    sys = t + sampleTime;
    function sys=mdlTerminate(t,x,u)
    sys = [];
```

9.3.2 编写方法

本小节主要针对 9.3.1 节中模板的各部分加以详细解释。编写 S-函数就是根据需求，用相应的代码去代替模板中对应部分的代码。

1）函数声明

第一行“function[sys,x0,str,ts] = sfuntmpl(t,x,u,flag)”是函数的声明，其中各参数的含义如表 9-2 所示。

表 9-2 参数含义

参 数 名	参 数 含 义
sfuntmpl	S-函数的名称，用户可换成期望的函数名
t	当前仿真时间
x	状态向量
u	输入向量
flag	用以标示 S-函数当前所处的仿真步骤，以便执行相应的回调函数，取值为 0、1、2、3、4 或 9，同时 5 保留以便扩展
sys	不同 flag 值，sys 返回值的含义不同： • flag=0，sys 返回系统描述 • flag=1，sys 返回微分结果 • flag=2，sys 返回更新结果 • flag=3，sys 返回输出结果 • flag=4，sys 返回下一个采样点 • flag=9，sys 返回空值
x0	只有在 flag=0 时返回初始状态值
str	只有在 flag=0 时返回状态字符串
ts	只有在 flag=0 时返回采样时间

同时完整的函数声明如下所示。

```
[sys,x0,str,ts] = sfunc(t,x,u,flag,p1,…,pn)
```

其中 p1,…,pn 为 S-函数模块的参数。

2）回调函数调用

在仿真过程中，Simulink 重复地调用 S-函数，并根据不同仿真步骤为参数 flag 赋予不同的值，以便调用指定的回调函数。表 9-3 列出了参数 flag 与 S-函数回调函数间的对应关系，以及各回调函数的说明。

表 9-3 参数 flag 与 S-函数回调函数的关系

flag 取值	S-函数回调函数	说　明
flag = 0	mdlInitializeSize	初始化
flag = 1	mdlDerivatives	计算微分
flag = 2	mdlUpdate	更新离散状态
flag = 3	mdlOutputs	计算输出
flag = 4	mdlGetTimeOfNextVarHit	计算下一个采样时间
flag = 9	mdlTerminate	结束仿真

语句“DAStudio.error(‘Simulink:blocks:unhandledFlag’,num2str(flag));”用于在 S-函数运行出错时显示报错信息。

3）回调函数 mdlInitializeSizes

回调函数 mdlInitializeSizes 中的 Sizes 是一个结构体，其各字段的含义如表 9-4 所示。

表 9-4 Sizes 结构体的各字段含义

字　段　名	含　义
sizes.NumConStates	连续状态的个数
sizes.NumDiscStates	离散状态的个数
sizes.NumOutputs	输出的数目（所有输出向量的宽度之和）
sizes.NumInputs	输入的数目（所有输入向量的宽度之和）
sizes.DirFeedthrough	有无直接馈入，等于 0 表示没有，等于 1 表示有
sizes.NumSampleTimes	采样时间的个数，至少是一个采样时间

其中，直接馈入是指系统的输出或可变采样时间是否受到输入的控制，如 y = ku(u 是输入，k 是放大因子，y 是输出)是直接馈入的，y = kx(x 是状态，k 是放大因子，y 是输出)不是直接馈入的。

下述语句的含义是，x0 表示状态的初始值，str=[]始终是空矩阵，ts=[00]采样周期设为 0 表示是连续系统，其中 n 为采样时间的个数，每个采样时间的设置如表 9-5 所示。

```
x0 = [];
str = [];
ts = [0 0];
```



表 9-5　ts 每个采样时间的设置

取　　值	含　　义
[0　0]	连续采样时间
[0　1]	连续采样时间，以最小步长运算
[period offset]	定步长离散采样时间，period 为周期，offset 为偏置量，且 0<offset<period，此时第 m 步的采样时间 Tm = m * period + offset
[-2　0]	变步长离散采样时间，采样时间由 flag = 4 的回调函数设置
[-1　0]	在最小步长内改变的函数并且采用驱动模块的采样时间
[-1　1]	在最小步长内不改变的函数并且采用驱动模块的采样时间

4）回调函数 mdlDerivatives

下述语句的含义：sys 表示状态的导数 $\dot{x}$，如 $x=(x_1x_2x_3...x_n)^T$，则 $sys=(sys(1)sys(2)...sys(n))^T$，其中，$sys(i)=\dot{x}$　$(i=1,2,\ldots,n)$；等号右边的“[]”表示导数的表达式。

```
sys = []
```

5）回调函数 mdlUpdate

下述语句的含义：sys 表示状态的下一步取值 x(n+1)，同理 sys 可以为向量；等号右边的“[]”表示下一步状态值的表达式。

```
sys = [];
```

6）回调函数 mdlOutputs

下述语句的含义：sys 表示输出 y，同理 sys 可以为向量；等号右边的“[]”表示输出的表达式。

```
sys = [];
```

7）回调函数 mdlGetTimeOfNextVarHit

下述语句的含义，sampleTime 表示采样步长，sys 表示下一步的采样时间，t+sampleTime 的位置为下一步采样时间的表达式，这里表示下一步的采样时间为在当前采样时间的基础上增加 1 秒。

```
sampleTime = 1;
sys = t+sampleTime;
```

8）回调函数 mdlTerminate

下述语句表示 sys 输出为空值。

```
sys = [];
```

9.3.3　实例

例 9-1 给出了 Level-1 M 文件型 S-函数的简单实例，下面通过不同类型的实例进一步说明 Level-1 M 文件型 S-函数的编写方法和编写技巧。

1．状态、输入和输出多维的情况

例 9-1 是状态、输入和输出都是一维的例子，例 9-2 给出一个状态、输入和输出都是多维的例子。

【例 9-1】 用 Level-1 M 文件型 S-函数（MySfunction.m）描述方程 $\begin{cases} \dot{x} = x\cos(tx+u) \\ x(0)=1 \\ y=3x \end{cases}$，其内容如下。

```
function [sys,x0,str,ts] = MySfunction1(t,x,u,flag)
switch flag,
     case 0,
       [sys,x0,str,ts]=mdlInitializeSizes;
case 1,
       sys=mdlDerivatives(t,x,u);
     case 3,
       sys=mdlOutputs(t,x,u);
  case {2, 4, 9 }
       sys = [];
     otherwise
       error(['Unhandled flag = ',num2str(flag)]);
end
function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates     = 1;          %表示仅一个连续状态 x
sizes.NumDiscStates     = 0;
sizes.NumOutputs        = 1;          %表示仅一个输出 x
sizes.NumInputs         = 1;          %表示仅一个输入 u
sizes.DirFeedthrough    = 0;
sizes.NumSampleTimes    = 0;
sys = simsizes(sizes);
x0  = 1;                              %表示 x(0)=1
str = [];
ts  = [];
function sys=mdlDerivatives(t,x,u)
sys=x*cos(t*x)+u;                     %表示 x 导数的表达式
function sys=mdlOutputs(t,x,u)
sys = 3*x;                            %表示输出 y=3*x
```

在 Simulink 中，使用该 S-函数的实例如图 9-6 所示，对于指定输入 u，计算状态 x(t)的轨迹并输出，运行后双击 Scope 模块可得到如图 9-7 所示的结果。

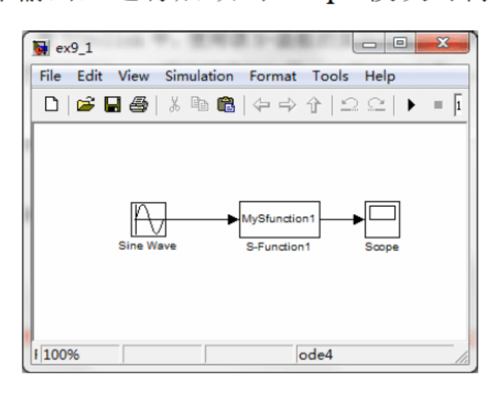


图 9-6　S-函数应用实例 1

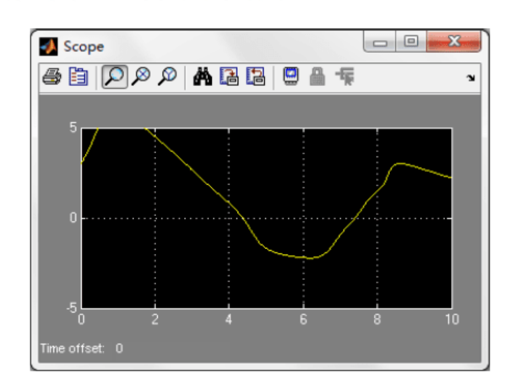


图 9-7　S-函数应用实例 1 运行结果

【例 9-2】 用 Level-1 M 文件型 S-函数（MySfunction2.m）描述方程

$$\begin{cases} \dot{x}_1 = 2 * x_2 \\ \dot{x}_2 = 3 * x_3 + u_1 \\ \dot{x}_3 = \dot{x}_2 \cos(tx_2) x_3 u_2 \\ x_1(0) = 1, x_2(0) = 1, x_3(0) = 2 \\ y = [x_1 2 * x_2 x_1 + x_3 x_1 + 2 * x_2] \end{cases}$$

，其内容如下。

```
function [sys,x0,str,ts] = MySfunction2(t,x,u,flag)
switch flag,
     case 0,
       [sys,x0,str,ts]=mdlInitializeSizes;
     case 1,
       sys=mdlDerivatives(t,x,u);
     case 2,
       sys=mdlUpdate(t,x,u);
     case 3,
       sys=mdlOutputs(t,x,u);
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
     case 9,
       sys=mdlTerminate(t,x,u);
  otherwise
       DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
     end

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 3;           %3 个状态
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 4;           %4 个输出
sizes.NumInputs      = 2;           %2 个输入
sizes.DirFeedthrough = 0;           %无直接馈入
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0 = [1;0;2];                       %初始条件[1 0 2]
str = [];
ts = [0 0];                         %连续采样时间

function sys=mdlDerivatives(t,x,u) %描述微分方程
sys(1) =x(2);                       %等价于 dx1=x2
sys(2) =x(3)+u(1);                  %等价于 dx2=x3+u1
sys(3) =x(1)^2*exp(-x(1))*sin(t*x(2))*x(3)*u(2);
                                 %等价于 dx3=x1^2*exp(-x1)*sin(t*x2)*x3*u2

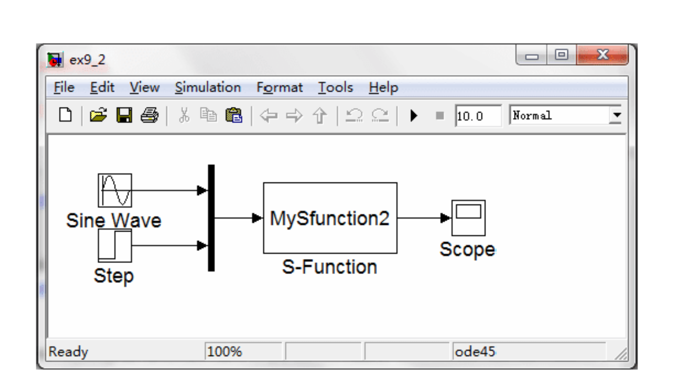
function sys=mdlUpdate(t,x,u)
sys = [];

function sys=mdlOutputs(t,x,u)      %描述输出方程
sys(1) = x(1);                      %等价于 y1=x1
```

```
    sys(2) = x(2);                              %等价于 y2=x2
    sys(3) = x(3);                              %等价于 y3=x3
    sys(4) = x(1)+x(2)+x(3);                    %等价于 y4=x1+x2+x3

    function sys=mdlGetTimeOfNextVarHit(t,x,u)  %将输出置空
    sys = [];
    function sys=mdlTerminate(t,x,u)
    sys = [];
```

在 Simulink 中，使用该 S-函数的实例如图 9-8 所示，即对于给定输入 u 计算输出 y，运行后双击 Scope 模块可以得到如图 9-9 所示的结果。



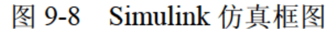


图 9-8 Simulink 仿真框图

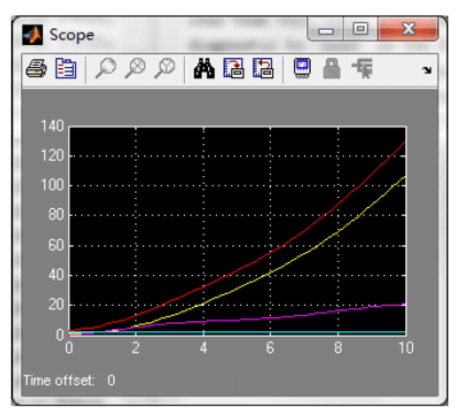


图 9-9 包含 4 个输入出的仿真曲线

由于输入是二维的，在图 9-8 中使用了一个 Mux 模块将两个一维输入联合后作为 S-函数的输入；如果信号本身是二维的，可以直接连接并作为 S-函数的输入；如果信号本身是大于二维的，必须经过适当处理，保证 S-函数的输入是且必须是二维的。

由于输出是 4 维的，所以可以得到如图 9-9 所示的 4 条输出曲线。

需要说明的是，S-函数保存的文件名和函数名要一致，保存 M-文件 S-函数的目录要包含在 MATLAB 的搜索路径文件之中。

2. 采用直接馈入的方式

例 9-3 实现状态空间的表示，并利用了直接馈入的方式。

【例 9-3】 用 Level-1 M 文件型 S-函数（MySfunction3.m）描述方程 $\begin{cases} \dot{X} = AX + Bu \\ X(0) = 0 \\ Y = CX + Du \end{cases}$（其中 $A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -1 & 1 & 4 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, C = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, D = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$，它等价于 $\begin{cases} \dot{x}_1 = x_1 \\ \dot{x}_2 = x_2 \\ \dot{x}_3 = -x_1 + x_2 + 4x_3 + u \\ x_1(0) = 0, x_2(0) = 0, x_3(0) = 0 \\ y_1 = 2x_1 + u \\ y_2 = 2x_2 + u \end{cases}$），

其内容如下。

```
function [sys,x0,str,ts] = MySfunction3(t,x,u,flag)
switch flag,
     case 0,
       [sys,x0,str,ts]=mdlInitializeSizes;
     case 1,
       sys=mdlDerivatives(t,x,u);
     case 2,
       sys=mdlUpdate(t,x,u);
     case 3,
       sys=mdlOutputs(t,x,u);
     case 4,
       sys=mdlGetTimeOfNextVarHit(t,x,u);
     case 9,
       sys=mdlTerminate(t,x,u);
     otherwise
       DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
end

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 3;          %三个状态
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;          %2 个输出
sizes.NumInputs      = 1;          %1 个输入
sizes.DirFeedthrough = 1;          %有直接馈入
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [0;0;0];                     %初始条件[0 0 0]
str = [];
ts = [0 0];                        %连续采样时间

function sys=mdlDerivatives(t,x,u) %描述微分方程
%紧凑描述
A=[1 0 0;0 1 0;-1 1 4];
B=[0;0;1] ;
sys=A*x+B*u;                       %等价于 dX=AX+Bu
%%展开描述
%sys(1) =x(1);                     %等价于 dx1=x1
%%sys(2) =x(2);                    %等价于 dx2=x2
%%sys(3) =-x(1)+x(2)+4 *x(3)+u;    %等价于 dx3=-x1+x2+4*x3+u

function sys=mdlUpdate(t,x,u)
sys = [];

function sys=mdlOutputs(t,x,u)     %描述输出方程
%紧凑描述
C=[2 0 0;0 1 0];
D=[1;0];
sys=C*x+D*u;                       %等价于 Y=CX+Du
%%展开描述
%sys(1) =2*x(1)+u;                 %等价于 y1=2*x1+u
```

```
function sys=mdlGetTimeOfNextVarHit(t,x,u)      %将输出置空
sys = [];

function sys=mdlTerminate(t,x,u)
sys = [];
```

在 Simulink 中，使用该 S-函数的实例如图 9-10 所示，运行后双击 Scope 模块可得到如图 9-11 所示的结果。

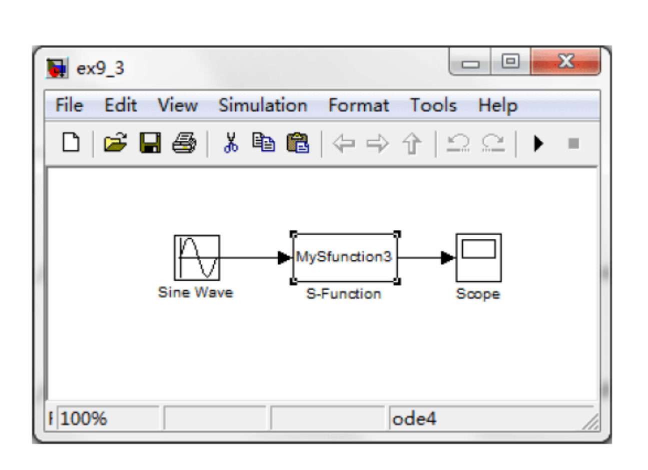


图 9-10　Simulink 仿真框图

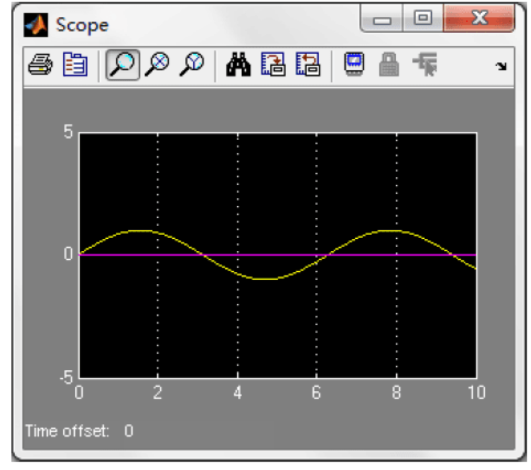


图 9-11　Simulink 仿真运行结果

本例中的语句 sizes.DirFeedthrough = 1；若改为 sizes.DirFeedthrough = 0；则系统会报错，因为回调函数 mdlOutputs 中利用了输出的信息，而 sizes.DirFeedthrough = 0；表示无直接馈入。

同时本例中给出了方程的两种表示方式，这两种表示方式是等价的，不仅是数学描述上等价，运行结果也是相同的。

3. 参数传递

前面的例子中并不涉及参数传递的问题，包括 S-函数本身的参数传递，以及 S-函数与其回调函数间的参数传递。下面的例子说明了这两类函数参数的传递方法。

【例 9-4】 用 Level-M 文件型 S-函数（MySfunction4.m）描述方程 $\begin{cases}\dot{X}=AX+Bu\\ X(0)=0\\ Y=CX+Du\end{cases}$（其中 $A=\begin{bmatrix}1&0&0\\0&1&0\\w1&w2&w3\end{bmatrix}, B=\begin{bmatrix}0\\0\\1\end{bmatrix}, C=\begin{bmatrix}1&0&0\\0&1&0\end{bmatrix}, D=\begin{bmatrix}0\\1\end{bmatrix}, W=\begin{pmatrix}w1&w2&w3\end{pmatrix}$），其内容如下。

```
function [sys,x0,str,ts] = MySfunction4(t,x,u,flag,P,q)
A=[0 1 0;0 1 0;P(1) P(2) P(3)];     %使用到外部参数向量 P，对应本例中的向量 W
B=[0;0;1] ;
C=[1 0 0;0 1 0];
D=[1;q];                            %其中使用到外部参数 q
```

```
switch flag,
     case 0,
       [sys,x0,str,ts]=mdlInitializeSizes(P,q);
                                             %附加外部参数，即使不使用也需要附加
     case 1,
       sys=mdlDerivatives(t,x,u,A,B);        %附加使用到的参数
     case 2,
       sys=mdlUpdate(t,x,u);
     case 3,
       sys=mdlOutputs(t,x,u,C,D);            %附加使用到的参数
     case 4,
       sys=mdlGetTimeOfNextVarHit(t,x,u);
     case 9,
       sys=mdlTerminate(t,x,u);
     otherwise
       DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
   end

function [sys,x0,str,ts]=mdlInitializeSizes(P,q); %附加外部参数
sizes = simsizes;
sizes.NumContStates  = 3;               %3 个状态
sizes.NumDiscStates  = 0;
sizes.NumOutputs     = 2;               %2 个输出
sizes.NumInputs      = 1;               %1 个输入
sizes.DirFeedthrough = 1;               %有直接馈入
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [0;0;0];                          %初始条件[0 0 0]'
str = [];
ts = [0 0];                             %连续采样时间

function sys=mdlDerivatives(t,x,u,A,B) %附加使用到的参数，描述微分方程
sys=A*x+B*u;                            %等价于 dX=AX+Bu

function sys=mdlUpdate(t,x,u)
sys = [];

function sys=mdlOutputs(t,x,u,C,D)     %附加使用到的参数，描述输出方程
sys=C*x+D*u;                            %等价于 Y=CX+Du

function sys=mdlGetTimeOfNextVarHit(t,x,u)  %将输出置空
sys = [];

function sys=mdlTerminate(t,x,u)
sys = [];
```

在 Simulink 中，使用该 S-函数的实例如图 9-12 所示。当 W=(w1 w2 w3)=(-1 -2 -3)且 q = 1 时，双击 S-Function 模块，如图 9-13 所示设置 S-Function parameters 属性即可，运行后双击 Scope 模块可得到如图 9-14 所示的结果。

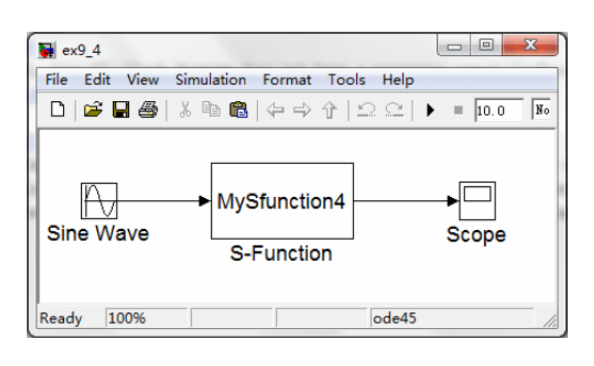


图 9-12　Simulink 仿真框图

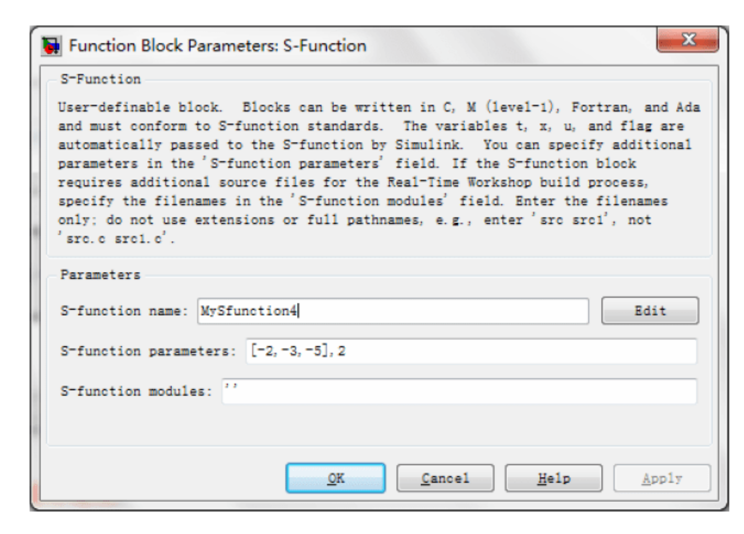


图 9-13　S-函数参数设置

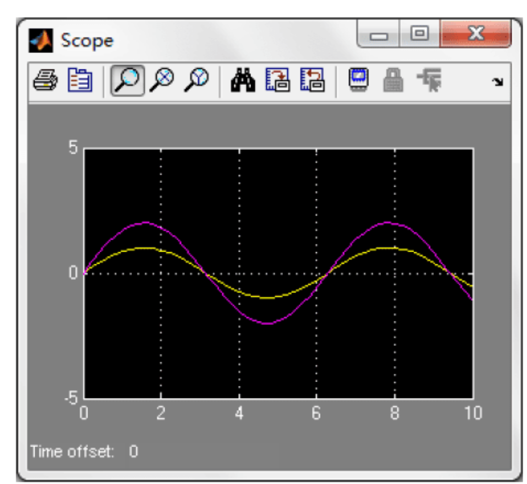


图 9-14　Simulink 仿真运行结果

需要说明如下两点。

（1）对于外部参数传递，多个参数的写法是若其中包含参数向量或矩阵，对应参数可以通过中括号对表示成相应的格式。

（2）对于内部参数传递，回调函数的声明和书写中都要附加被传递的参数名。

【例 9-5】 用 Level-1 M 文件型 S-函数（MySfunction5.m）描述方程 $\begin{cases} X(n+1)=AX(n)+Bu(n) \\ X(0)=0 \\ Y(n)=CX(n)+Du(n) \end{cases}$（其中 $A=\begin{bmatrix} 0 & -1 & 0 \\ 1 & 1 & -1 \\ 1 & 3 & 3 \end{bmatrix}, B=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, C=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, D=\begin{bmatrix} 1 \\ 2 \end{bmatrix}$），其内容如下。

```
function [sys,x0,str,ts] = MySfunction5(t,x,u,flag)
switch flag,
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes;
  case 1,
    sys = mdlDerivatives(t,x,u);
  case 2,
    sys = mdlUpdate(t,x,u);
  case 3,
    sys = mdlOutputs(t,x,u);
  case 4,
    sys = mdlGetTimeOfNextVarHit(t,x,u);
  case 9,
```

```
    sys = mdlTerminate(t,x,u);
  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
end

function [sys,x0,str,ts]=mdlInitializeSizes
sizes = simsizes;
sizes.NumContStates  = 0;
sizes.NumDiscStates  = 3;         %3 个离散状态
sizes.NumOutputs     = 2;         %2 个输出
sizes.NumInputs      = 1;         %1 个输入
sizes.DirFeedthrough = 1;         %有直接馈入
sizes.NumSampleTimes = 1;
sys = simsizes(sizes);
x0  = [0;0;0];                    %初始条件[0  0  0]
str = [];
ts = [0.1 0];                     %定步长离散采样时间，采样周期为 0.1，偏置量为 0

function sys=mdlDerivatives(t,x,u)
sys=[];

function sys=mdlUpdate(t,x,u)     %描述差分方程
A=[0 -1 0;0 0 -1; 1 3 3];
B=[0;0;1] ;
sys=A*x+B*u;                      %等价于 X(n+1)=AX(n)+Bu(n)

function sys=mdlOutputs(t,x,u)    %描述输出方程
C=[1 0 0;0 1 0];
D=[1;2];
sys=C*x+D*u;                      %等价于 Y(n)=CX(n)+Du(n)

function sys=mdlGetTimeOfNextVarHit(t,x,u)  %将输出置空
sys = [];

function sys=mdlTerminate(t,x,u)
sys = [];
```

在 Simulink 中，使用该 S-函数的实例如图 9-15 所示，运行后双击 Scope 模块可得到如图 9-16 所示的结果。

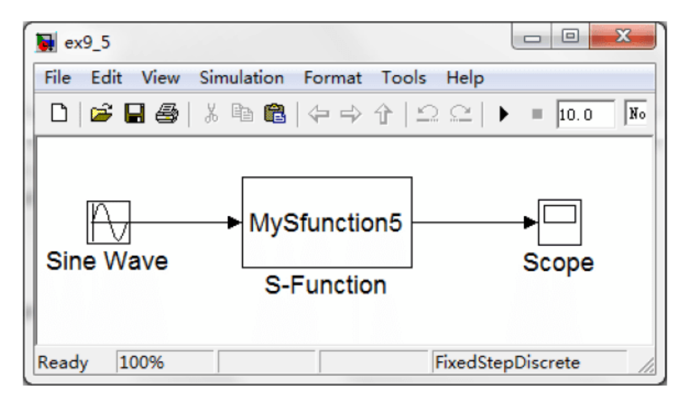


图 9-15　Simulink 仿真框图

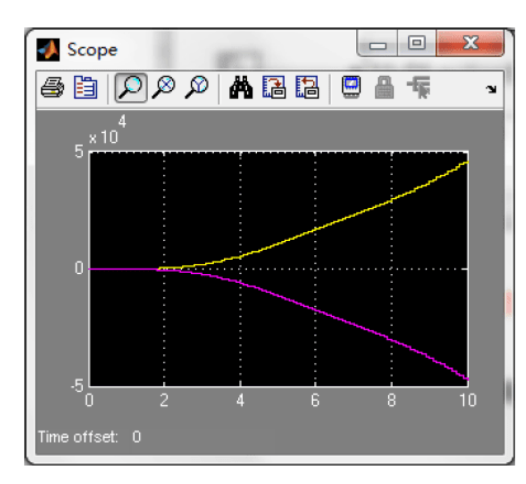


图 9-16　Simulink 仿真运行结果

本例中给定了采样周期及偏置量，仿真结果也显示采样周期为 0.1s，偏置量为 0。

5. 混合系统

混合系统指既有连续又有离散状态的系统，下面给出一个混合系统的例子，其中还涉及多个仿真时间的处理。

【例 9-6】 用 Level-1 M 文件型 S-函数（MySfunction6.m）描述方程

$$\begin{cases} \dot{x}_1 = 7*u \\ x_2(n+1) = x_2(n) + x_1(T=1) \\ x_1(0) = 0, x_2 = 0 \\ y = [x_1 \quad x_2]^T \end{cases}$$

，其内容如下。

```
function [sys,x0,str,ts] = MySfunction6(t,x,u,flag)
dperiod = 1;             %离散状态的采样周期
doffset = 0;             %离散状态的采样时间偏置量
switch flag,
  case 0,
    [sys,x0,str,ts]=mdlInitializeSizes(dperiod,doffset);  %附加使用到的参数
  case 1,
    sys=mdlDerivatives(t,x,u);
  case 2,
    sys=mdlUpdate(t,x,u,dperiod,doffset);                  %附加使用到的参数
  case 3,
    sys=mdlOutputs(t,x,u);
  case 4,
    sys=mdlGetTimeOfNextVarHit(t,x,u);
  case 9,
    sys=mdlTerminate(t,x,u);
  otherwise
    DAStudio.error('Simulink:blocks:unhandledFlag', num2str(flag));
end

function [sys,x0,str,ts]=mdlInitializeSizes(dperiod,doffset)
 %附加使用到的参数
sizes = simsizes;
sizes.NumContStates  = 1;                 %1 个连续状态
sizes.NumDiscStates  = 1;                 %1 个离散状态
sizes.NumOutputs      = 2;                %2 个输出
sizes.NumInputs       = 1;                %1 个输入
sizes.DirFeedthrough = 0;                 %无直接馈入
sizes.NumSampleTimes = 2;                 %2 个采样时间
sys = simsizes(sizes);
x0  = [0;0];                              %初始条件[0 0]
str = [];
ts = [0 0;                                %两个采样时间，一个是连续采样时间
dperiod,doffset];                         %一个是采样周期为 1，偏置量为 0

function sys=mdlDerivatives(t,x,u)        %描述微分方程
sys=7*u;                                  %等价于 dx=7u
```

```
function sys=mdlUpdate(t,x,u,dperiod,doffset)
    %附加使用到的参数，描述差分方程

t times=(t-doffset)/dperiod;
%计算当前时间扣除偏置量后相对采样周期的倍数
t err= abs(round(t times)-t times)          %比较与正倍数的距离
if t err < 1e-8
 %距离足够小则认为时刻位于采样点上，此时修正 x(n+1)，否则 x(n+1)值不变
    sys =x(2)+ x(1);                        %等价于 x(n+1)=x(n)+x1
else
sys = [];
end

function sys=mdlOutputs(t,x,u)              %描述输出方程
sys=x;                                      %等价于 y=[x1;x2]

function sys=mdlGetTimeOfNextVarHit(t,x,u) %将输出置空
sys = [];

function sys=mdlTerminate(t,x,u)
sys = [];
```

在 Simulink 中，使用该 S-函数的实例如图 9-17 所示，运行后双击 Scope 模块可得到如图 9-18 所示的结果。

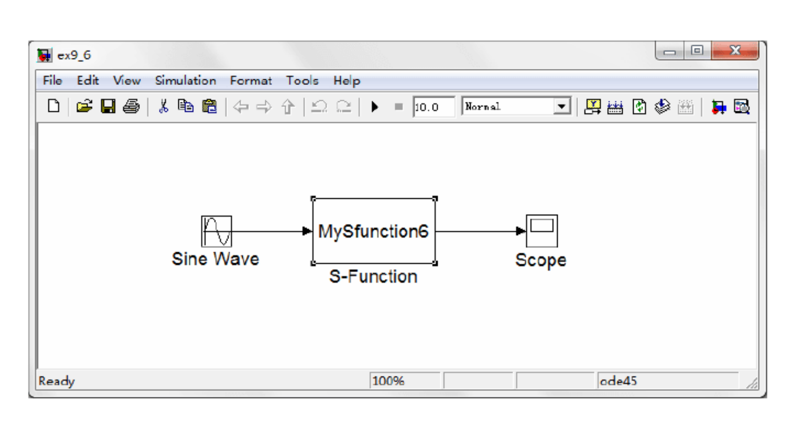


图 9-17 Simulink 仿真框图

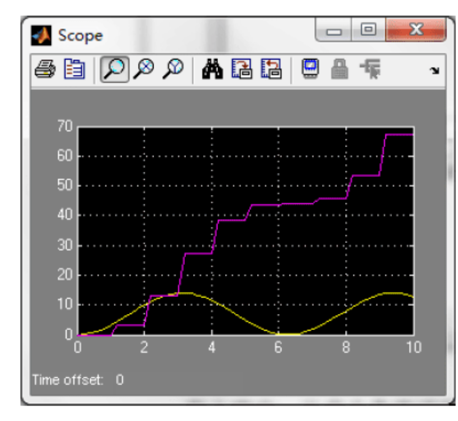


图 9-18 Simulink 仿真运行结果

由本例可以看出，处理混合系统时，需要对离散状态的更新做特殊处理，令其在指定时间点上操作，其他时刻不更新。

9.4 Level-2 M 文件型

在上节中，从多个角度详细介绍了 Level-1 M 文件型 S-函数的编写方法，因为这是当前应用最广，用户使用最数量的方式。目前，MATLAB 在保留 Level-1 M 文件型 S-函数的基础上，力推 Level-2 M 文件型 S-函数，并提供了由 Level-1 M 文件型到 Level-2 M 文件型

S-函数的转换方法。本节将着重介绍 Level-2 M 文件型 S-函数。

9.4.1 概述

与查看 Level-1M 文件型 S-函数演示程序的过程类似，可以通过操作看到如图 9-4 所示的界面，继续双击“Level-2 M-files”，可以看到如图 9-19 所示的界面，其中列出了 Level-2 M 文件型 S-函数的大量实例。

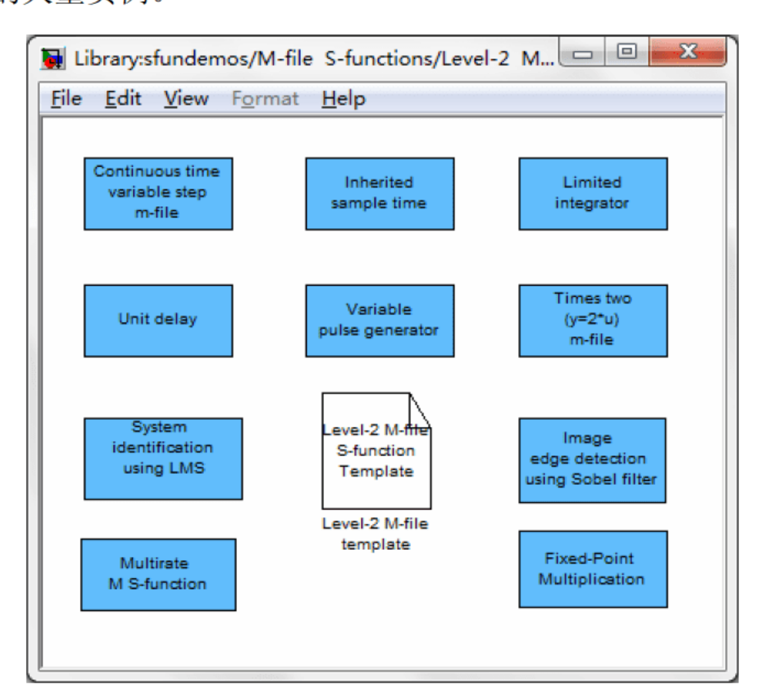


图 9-19　Level-2 M 文件型 S-函数实例演示模块

图 9-19 中包含一个 Level-2 M-files S-function Template 示例，它提供了书写该类型 S-函数的模板，删除注释后的具体内容如下。

```
function msfuntmpl(block)
setup(block);

function setup(block)
  block.NumInputPorts  = 1;
  block.NumOutputPorts = 1;
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;
  block.InputPort(1).DatatypeID  = 0;
  block.InputPort(1).Complexity  = 'Real';
  block.OutputPort(1).DatatypeID  = 0;
  block.OutputPort(1).Complexity  = 'Real';
  block.NumDialogPrms     = 3;
  block.DialogPrmsTunable = {'Tunable','Nontunable','SimOnlyTunable'};
  block.SampleTimes = [0 0];
  block.SetAccelRunOnTLC(false);
  block.SimStateCompliance = 'DefaultSimState';
```

```
  block.RegBlockMethod('CheckParameters', @CheckPrms);
  block.RegBlockMethod('SetInputPortSamplingMode', @SetInpPortFrameData);
  block.RegBlockMethod('SetInputPortDimensions', @SetInpPortDims);
  block.RegBlockMethod('SetOutputPortDimensions', @SetOutPortDims);
  block.RegBlockMethod('SetInputPortDataType', @SetInpPortDataType);
  block.RegBlockMethod('SetOutputPortDataType', @SetOutPortDataType);
  block.RegBlockMethod('SetInputPortComplexSignal', @SetInpPortComplexSig);
  block.RegBlockMethod('SetOutputPortComplexSignal', @SetOutPortComplexSig);
  block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
  block.RegBlockMethod('ProcessParameters', @ProcessPrms);
  block.RegBlockMethod('InitializeConditions', @InitializeConditions);
  block.RegBlockMethod('Start', @Start);
  block.RegBlockMethod('Outputs', @Outputs);
  block.RegBlockMethod('Update', @Update);
  block.RegBlockMethod('Derivatives', @Derivatives);
  block.RegBlockMethod('Projection', @Projection);
  block.RegBlockMethod('SimStatusChange', @SimStatusChange);
  block.RegBlockMethod('Terminate', @Terminate);
  block.RegBlockMethod('WriteRTW', @WriteRTW);
function CheckPrms(block)
  a = block.DialogPrm(1).Data;
  if ~strcmp(class(a), 'double')
    DAStudio.error('Simulink:block:invalidParameter');
  end

function ProcessPrms(block)
  block.AutoUpdateRuntimePrms;

function SetInpPortFrameData(block, idx, fd)
  block.InputPort(idx).SamplingMode = fd;
  block.OutputPort(1).SamplingMode  = fd;

function SetInpPortDims(block, idx, di)
  block.InputPort(idx).Dimensions = di;
  block.OutputPort(1).Dimensions = di;

function SetOutPortDims(block, idx, di)
  block.OutputPort(idx).Dimensions = di;
  block.InputPort(1).Dimensions   = di;

function SetInpPortDataType(block, idx, dt)
  block.InputPort(idx).DataTypeID = dt;
  block.OutputPort(1).DataTypeID = dt;

function SetOutPortDataType(block, idx, dt)
  block.OutputPort(idx).DataTypeID  = dt;
  block.InputPort(1).DataTypeID     = dt;

function SetInpPortComplexSig(block, idx, c)
  block.InputPort(idx).Complexity = c;
  block.OutputPort(1).Complexity  = c;
```

```
function SetOutPortComplexSig(block, idx, c)
  block.OutputPort(idx).Complexity = c;
  block.InputPort(1).Complexity   = c;

function DoPostPropSetup(block)
  block.NumDworks = 1;
  block.Dwork(1).Name            = 'x1';
  block.Dwork(1).Dimensions      = 1;
  block.Dwork(1).DatatypeID      = 0;
  block.Dwork(1).Complexity      = 'Real';
  block.Dwork(1).UsedAsDiscState = true;
  block.AutoRegRuntimePrms;

function InitializeConditions(block)

function Start(block)
   block.Dwork(1).Data = 0;

function WriteRTW(block)
   block.WriteRTWParam('matrix', 'M',[1 2; 3 4]);
   block.WriteRTWParam('string', 'Mode', 'Auto');

function Outputs(block)
    block.OutputPort(1).Data= block.Dwork(1).Data + block.InputPort(1).Data;

function Update(block)
    block.Dwork(1).Data = block.InputPort(1).Data;

function Derivatives(block)

function Projection(block)

function SimStatusChange(block, s)
    if s == 0
    disp('Pause has been called');
  elseif s == 1
    disp('Continue has been called');
  end
function Terminate(block)
```

9.4.2 编写方法

本小节主要针对 9.4.1 节中模板的各部分加以详细解释。编写 S-函数就是根据需求，用相应的代码去替代末班中对应部分的代码。

1. 两种 M 文件型 S-函数的关系

前面小节详细讲解了 Level-1 M 文件型 S-函数模板中的各部分，下面先介绍如何由 Level-1 M 文件型转换为 Level-2 M 文件型 S-函数。

首先，以 Level-2 M 文件型 S-函数的参数 flag 为基准，建立 Level-1 M 文件型和 Level-2 M 文件型 S-函数两个模板间的对应关系，如表 9-6 所示。

表 9-6　两种 M 文件型 S-函数模板间基于参数 flag 的对应关系

Level-1 M 文件型	Level-2 M 文件型
function[sys,x0,str,ts] = MySfunction(t,x,u,flag)	function MySfunction(block)
case 0,　[sys,x0,str,ts] = mdlInitializeSizes;	setup(block)
case 1, sys = mdlDerivatives(t,x,u);	block.RegBlockMethod('Derivatives',@Derivatives);
case 2, sys = mdlUpdate(t,x,u);	block.RegBlockMethod('Update',@Update);
case 3, sys = mdlOutputs(t,x,u);	block.RegBlockMethod('Outputs',@Output);
case 4, sys = mdlGetTimeOfNextVarHit(t,x,u);	模板中无对应语句设置，可通过如下回调函数实现： block.RegBlockMethod('Outputs',@Output)
case 9, sys = mdlTerminate(t,x,u);	block.RegBlockMethod('Terminate',@Terminate)

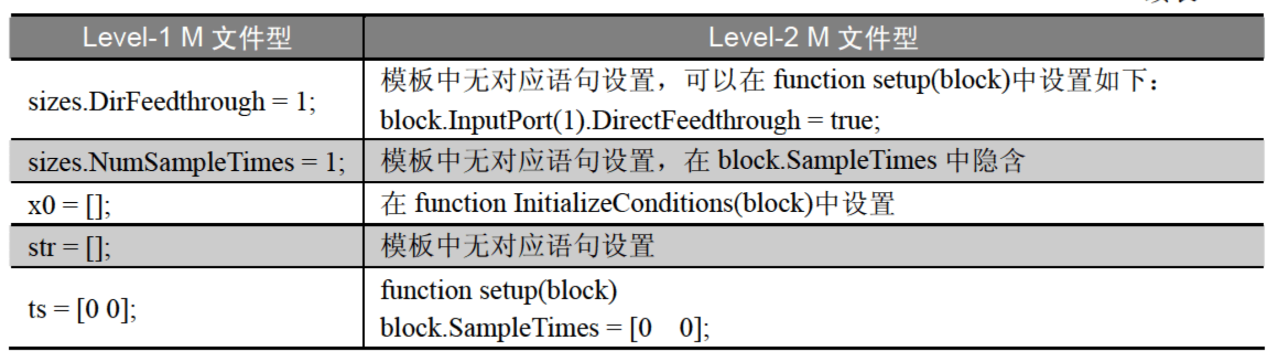
注意如下两点。

- Level-2 M 文件型 S-函数不支持过零检验。
- function[sys,x0,str,ts] = MySfunction(t,x,u,flag)中的参数 flag 在 Level-2 M 文件型 S-函数中已经不存在，参数 t 对应 block.CurrentTime，参数 x 对应 block.Dwork，参数 u 对应 block.InputPort.Data，参数 ts 对应 block.SampleTimes，参数 str 已经不存在，参数 x0 的内容将在后面讲解，参数 sys 在系统输出时对应 block.OutputPort.Data。
- 其次，以 Level-1 M 文件型 S-函数的回调函数 mdlInitializeSizes 为基准，建立 Level-1 M 文件型和 Level-2 M 文件型 S-函数两个模板间的对应关系，如表 9-7 所示。

表 9-7　两种 M 文件型 S-函数模板间基于回调函数 mdlInitializeSizes 的对应关系

Level-1 M 文件型	Level-2 M 文件型
sizes.NumContStates = 0;	function DoPostPropSetup(block) block.NumContStates = 1; %声明连续状态数 block.AutoRegRuntimePrms; %将所有可调参数注册为运行参数
sizes.NumDiscStates = 0;	function DoPostPropSetup(block) block.NumDworks = 1; %声明离散状态数 block.Dwork(1).Name = 'x1'; %状态名 block.Dwork(1).Dimensions = 1; %状态维数 block.Dwork(1).DatatypeID = 0; %状态数据类型，0 为 double 型 block.Dwork(1).Complexity = 'Real'; %状态数据类型，'Real'为实数 block.Dwork(1).UsedAsDiscState = true; %状态类型，true 为离散，false 为连续 block.AutoRegRuntimePrms; %将所有可调参数注册为运行参数
sizes.NumOutputs = 0;	function setup(block)
sizes.NumInputs = 0;	block.NumInputPorts = 1; %设置输入端口数，每个端口可包含多个输入 block.NumOutputPorts = 1; %设置输出端口数，每个端口可包含多个输出 block.InputPort(1).DatatypeID = 0; %第一个输入端口数据类型，double 型 block.InputPort(1).Complexity = 'Real' %数据类型，实数型 block.OutputPort(1).DatatypeID = 0; %第一个输出端口数据类型，double 型 block.OutputPort(1).Complexity = 'Real'; %数据类型，实数型 模板中无对应语句设置端口的输入数目，可以在 function setup(block)中如下设置: block.InputPort(1).Dimensions = 2; %第一个输入端口包含 2 个输入

续表



Level-1 M 文件型	Level-2 M 文件型
sizes.DirFeedthrough = 1;	模板中无对应语句设置，可以在 function setup(block)中设置如下： block.InputPort(1).DirectFeedthrough = true;
sizes.NumSampleTimes = 1;	模板中无对应语句设置，在 block.SampleTimes 中隐含
x0 = [];	在 function InitializeConditions(block)中设置
str = [];	模板中无对应语句设置
ts = [0 0];	function setup(block) block.SampleTimes = [0　0];

最后，以 Level-1 M 文件型 S-函数的回调函数为基准，建立 Level-1 M 文件型和 Level-2 M 文件型 S-函数两个模板间的对应关系，如表 9-8 所示。

表 9-8　两种 M 文件型 S-函数模板间基于回调函数的对应关系

Level-1 M 文件型	Level-2 M 文件型
function sys = mdlDerivatives(t,x,u) sys=[];	function Derivatives(block) block.Derivatives.Data = [];
function sys = mdlUpdate(t,x,u) sys=[];	function Update(block) block.Dwork.Data = [];
function sys = mdlOutputs(t,x,u) sys=[];	function Outputs(block) block.OutputPort.Data = [];
function sys = mdlGetTimeOfNextVarHit(t,x,u)	功能包含在 function Outputs(block)中
function sys = mdlTerminate(t,x,u)	function Terminate(block)

综上，按照 Level-1 M 文件型书写模板，给出了其中语句及参数与 Level-2 M 文件型 S-函数的对应关系，这为 Level-1 M 文件型 S-函数的移植提供了方法，同时也介绍了 Level-2 M 文件型 S-函数模板的有关内容。

2. Level-2 M 文件型 S-函数的特性

Level-2 M 文件型 S-函数模板中除了具有前面介绍的与 Level-1 M 文件型相同的属性外，还有其特有的属性。

首先，介绍回调函数 setup 中的新增语句，如表 9-9 所示。

表 9-9　回调函数 setup 中的新增语句

语　句	含　义

block.RegBlockMethod('CheckParameters',@CheckPrms);
block.RegBlockMethod('SetInputPortSamplingMode',@SetInpPortFrameData);
block.RegBlockMethod('SetInputPortDimensions',@SetInpPortDims);
block.RegBlockMethod('SetOutputPortDimensions',@SetOutPortDims);
block.RegBlockMethod('SetInputPortDataType',@SetInpPortDataType);
block.RegBlockMethod('SetOutputPortDataType',@SetOutPortDataType);
block.RegBlockMethod('SetInputPortComplexSignal',
@SetInpPortComplexSig);
block.RegBlockMethod('SetOutputPortComplexSignal',
@SetOutPortComplexSig);
block.RegBlockMethod('ProcessParameters',@ProcessPrms);
	新增回调函数声明

续表

语　　句	含　　义

block.RegBlockMethod('Start',@Start);
block.RegBlockMethod('Projection',@Projection);
block.RegBlockMethod('SimStatusChange',@SimStatusChanges);
block.RegBlockMethod('WriteRTW',@WriteRTW);
	新增回调函数声明

block.NumDialogPrms = 3; block.DialogPrmsTunable = {'Tunable','Nontunable','SimOnlyTunable'};	设置外部参数和参数类型（可调、不可调或仅仿真可调）
block.SetPreCompInpPortInfoToDynamic; block.SetPreCompOutPortInfoToDynamic;	设置输入/输出端口的属性为继承性或动态
block.SetAccelRunOnTLC(false);	指定是否使用 TLC 文件生成仿真目标文件

其次，介绍新增的回调函数，如表 9-10 所示。

表 9-10　新增的回调函数

名　　称	含　　义
function CheckPrms(block)	检查参数的合法性
function ProcessPrms(block)	更改运行参数值
function SetInpPortFrameData(block,idx,fd)	检查和设置输入端口属性
function SetInpPortDims(block,idx,di)	检查和设置输入端口维数
function SetOutPortDims(block,idx,di)	检查和设置输出端口维数
function SetInpPortDataType(block,idx,dt)	检查和设置输入端口数据类型
function SetOutPortDataType(block,idx,dt)	检查和设置输出端口数据类型
function SetInpPortComplexSig(block,idx,c)	检查和设置输入端口是否为复数属性
function SetOutPortComplexSig(block,idx,c)	检查和设置输出端口是否为复数属性
function Start(block)	初始化状态和工作空间变量值
function WriteRTW(block)	向 RTW 文件写入指定信息
function Projection(block)	仿真步中更新预测值
function SimStatusChange(block,s)	仿真暂停时（s=0)和重启时（s=1)

综上，详细介绍了 Level-2 M 文件型 S-函数模板的内容，这为 Level-2 M 文件型 S-函数的书写提供了便利。

9.4.3　实例

下面通过不同类型的实例进一步说明 Level-2 M 文件型 S-函数的编写方法和编写技巧。

1．连续系统

这里给出一个连续系统的实例。

【例 9-7】 用 Level-2 M 文件型 S-函数（MySfunction7.m）描述方程 $\begin{cases} X = AX + Bu \\ X(0) = 0 \\ Y = CX + Du \end{cases}$ （其

中 $A=\begin{bmatrix}0 & 1 & 0\\0 & 0 & 1\\-2 & -3 & -5\end{bmatrix}, B=\begin{bmatrix}0\\0\\1\end{bmatrix}, C=\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\end{bmatrix}, D=\begin{bmatrix}1\\2\end{bmatrix}$），其内容如下。

```
function MySfunction7 (block)
setup(block);
function setup(block)
  block.NumInputPorts  = 1;
  block.NumOutputPorts = 1;
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;
  block.InputPort(1).DatatypeID  = 0;  % double
  block.InputPort(1).Complexity  = 'Real';
  block.InputPort(1).Dimensions        = 1;
  block.InputPort(1).DirectFeedthrough = true; %有直接馈入
  block.OutputPort(1).DatatypeID  = 0; % double
  block.OutputPort(1).Complexity  = 'Real';
block.OutputPort(1).Dimensions = 2; %一个输出端口包含两个输出
  block.SampleTimes = [0 0];
  block.SetAccelRunOnTLC(false);
%使用到的回调函数
  block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
  block.RegBlockMethod('InitializeConditions', @InitializeConditions);
  block.RegBlockMethod('Outputs', @Outputs);
  block.RegBlockMethod('Derivatives', @Derivatives);

function DoPostPropSetup(block)
  block.NumContStates = 3;  %3 个连续状态
  block.AutoRegRuntimePrms;

function InitializeConditions(block)
block.ContStates.Data = [0;0;0];

function Outputs(block)
C = [1 0 0;0 1 0];
D = [1;2];
block.OutputPort(1).Data=
    C*block.ContStates.Data+D*block.InputPort(1).Data;

function Derivatives(block)
A=[0 1 0;0 1 0;-2 -3 -5];
B=[0;0;1] ;
block.Derivatives.Data=A*block.ContStates.Data+B* block.InputPort(1).Data;
```

在 Simulink 中，使用该 S-函数的实例如图 9-20 所示。

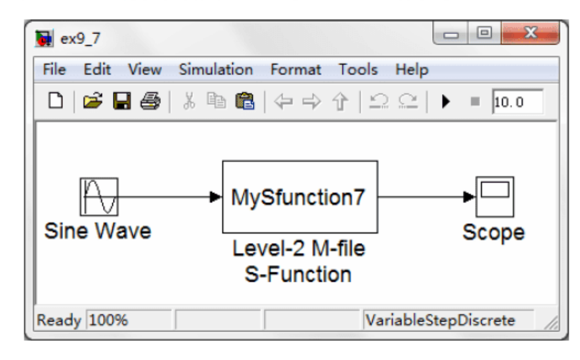


图 9-20　Simulink 仿真框图

本例给出的是针对连续状态的仿真，而模板给出的是针对离散状态的仿真。这里需要说明的是，离散状态与连续状态在表述上存在较大差异，在应用时要特别注意。

2．离散系统

这里给出一个离散系统的例子，同时将外部参数传入 S-函数。

【例 9-8】 用 Level-2 M 文件型 S-函数（MySFunction8.m）描述方程 $\begin{cases} X(n+1)=AX(n)+Bu(n) \\ X(0)=0 \\ Y(n)=CX(n)+Du(n) \end{cases}$（其中 $A=\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ e1 & e2 & e3 \end{bmatrix}, B=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, C=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, E=(e1 \quad e2 \quad e3)$），其内容如下。

```
function MySfunction8(block)
setup(block);

function setup(block)
  block.NumInputPorts  = 1;
  block.NumOutputPorts = 1;
  block.SetPreCompInpPortInfoToDynamic;
  block.SetPreCompOutPortInfoToDynamic;
  block.InputPort(1).DatatypeID  = 0;
  block.InputPort(1).Complexity  = 'Real';
  block.InputPort(1).Dimensions    = 1;
  block.InputPort(1).DirectFeedthrough = true; %有直接馈入
  block.OutputPort(1).DatatypeID  = 0; % double
  block.OutputPort(1).Complexity  = 'Real';
  block.OutputPort(1).Dimensions = 2;  %一个输出端口包含两个输出
  block.NumDialogPrms = 2;             %外部传入两个参数 E、q
  block.DialogPrmsTunable = {'Tunable', 'Tunable'};
  block.SampleTimes = [0.1 0];         %采样周期为 0.1，偏置量为 0
  block.SetAccelRunOnTLC(false);       %使用到的回调函数
  block.RegBlockMethod('PostPropagationSetup', @DoPostPropSetup);
  block.RegBlockMethod('InitializeConditions', @InitializeConditions);
  block.RegBlockMethod('Outputs', @Outputs);
  block.RegBlockMethod('Update', @Update);

function DoPostPropSetup(block)
  block.NumDworks = 3; %3 个离散状态
  block.Dwork(1).Name            = 'x1';   %第 1 个离散状态
  block.Dwork(1).Dimensions      = 1;
  block.Dwork(1).DatatypeID      = 0;      %double
  block.Dwork(1).Complexity      = 'Real'; %real
  block.Dwork(1).UsedAsDiscState = true;
  block.Dwork(2).Name            = 'x2';   %第 2 个离散状态
  block.Dwork(2).Dimensions      = 1;
  block.Dwork(2).DatatypeID      = 0;      %double
  block.Dwork(2).Complexity      = 'Real'; %real
  block.Dwork(2).UsedAsDiscState = true;
  block.Dwork(3).Name            = 'x3';   %第 3 个离散状态
  block.Dwork(3).Dimensions      = 1;
```

```
  block.Dwork(3).DatatypeID      = 0;      % double
  block.Dwork(3).Complexity      = 'Real'; % real
  block.Dwork(3).UsedAsDiscState = true;
  block.AutoRegRuntimePrms;

function InitializeConditions(block)
  block.Dwork(1).Data = 0; %为第 1 个离散状态赋值
  block.Dwork(2).Data = 0; %为第 2 个离散状态赋值
  block.Dwork(3).Data = 0; %为第 3 个离散状态赋值

function Outputs(block)
block.OutputPort(1).Data(1) =block.Dwork(1).Data + block.InputPort(1).Data;
block.OutputPort(1).Data(2)=block.Dwork(2).Data+
  block.DialogPrm(2).Data*block.InputPort(1).Data;
% block.DialogPrm(2).Data 表示第 2 个参数 q 的值
function Update(block)
block.Dwork(1).Data =-block.Dwork(2).Data;
block.Dwork(2).Data =-block.Dwork(3).Data;
block.Dwork(3).Data = block.DialogPrm(1).Data(1)* block.Dwork(1).Data+ ...
                      block.DialogPrm(1).Data(2)* block.Dwork(2).Data+ ...
block.DialogPrm(1).Data(3)* block.Dwork(3).Data+ block.InputPort(1).Data;
% block.DialogPrm(1).Data(1)表示第 1 个参数的第 1 个分量 e1 的值
```

在 Simulink 中，使用该 S-函数的实例如图 9-21 所示。当 $E=(e1\ e2\ e3)$和 $q=2$ 时，双击 S-Function 模块，如图 9-22 所示设置 S-function parameters 属性即可，且运行结果如图 9-23 所示。

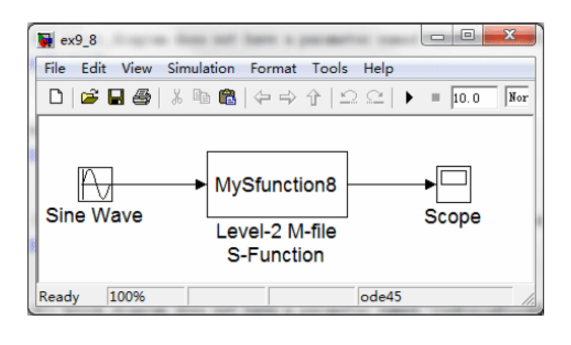


图 9-21 Simulink 仿真框图

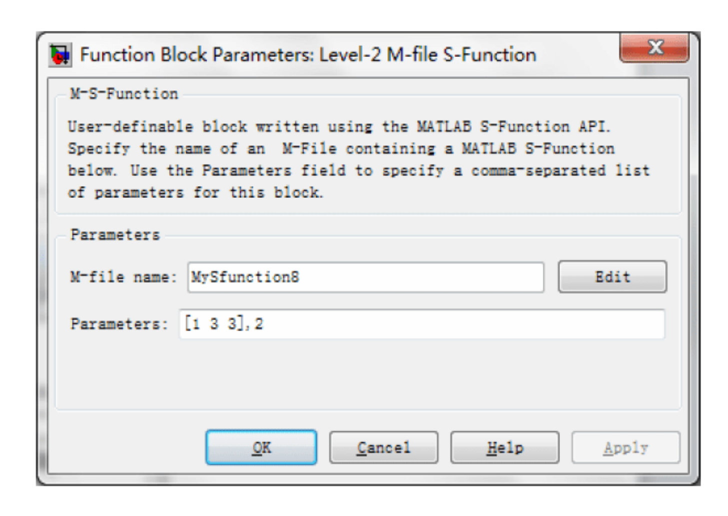


图 9-22 S-函数参数设置

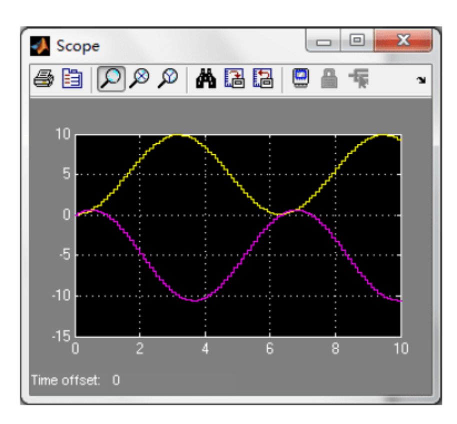


图 9-23 Simulink 仿真运行结果

本例给定了采样周期和偏置量，仿真结果也显示采样周期为 0.1s，偏置量为 0。

9.5 C MEX 文件型

前面详细介绍了两类 M 文件型 S-函数的编写方法，本节将重点介绍 C MEX 文件型 S-函数的编写方法。除了 C MEX 文件型，还有 C++ MEX、Fortran MEX 文件型 S-函数。在这些 MEX 文件型 S-函数中，C MEX 文件型是最常用的。

9.5.1 概述

1. MEX 文件简介

在 MATLAB 中，可调用的 C 或 Fortran 语言程序称为 MEX 文件，MATLAB 可以直接把 MEX 文件视为其内建函数进行调用。MEX 文件是动态链接的例程，MATLAB 解释器可以自动载入并执行它。MEX 文件主要有以下几方面的应用。

- ❑ 在 MATLAB 中，M 文件的计算速度特别是循环迭代的速度远比 C 语言慢，因此可以把要求大量循环迭代的部分用 C 语言编写为 MEX 文件，以提高计算速度。
- ❑ 对于已经开发的 C 语言程序，则不必将其转化为 M 文件而重复劳动，通过添加入口程序 mexFunction，即可由 MATLAB 调用。
- ❑ 直接控制硬件，如 A/D 采集卡、D/A 输出卡等，以用于数据采集或控制应用。

2. C MEX 文件简介

C MEX 文件，就是基于 C 语言编写的 MEX 文件，是 MATLAB 应用程序接口的一个重要组成部分。通过它不但可以将现有的使用 C 语言编写的函数轻松地引入 MATLAB 环境中使用，避免了重复的程序设计，而且可以使用 C 语言为 MATLAB 定制用于特定目的的函数，以完成在 MATLAB 中不易实现的任务，同时还可以使用 C 语言提高 MATLAB 环境中数据的处理效率。

C 语言的 MEX 文件的源程序由两个非常明显的部分组成。

- ❑ 计算程序，即在 MEX 文件中完成计算功能的程序代码。计算可以是普通的 C 语言程序，按照 C 语言规则编写即可。
- ❑ 入口程序，将计算程序与 MATLAB 连接的入口函数 mexFunction。

需要注意的是，MEX 文件虽然具有较强大的功能，但并不是对所有的应用都恰当。MATLAB 是一个高效率的编程系统，特别适合于工程计算、系统仿真等应用，其最大优点就是将人们从繁杂的程序中解放出来。因此，能够用 M 文件完成的程序，应尽量使用 MATLAB 编写，除非遇到必须使用 MEX 文件的情况。

与查看 Level-1 和 Level-2 M 文件型 S-函数演示程序的过程类似，可以通过操作看到如图 9-3 所示的界面，继续双击“C-files”，可以看到如图 9-24 所示的界面，其中列出了 C MEX 文件型 S-函数的大量实例。

图 9-24 中包含了一个 Basic C-MEX Template 和一个 Detailed C-MEX Template 示例，一个是为了基本应用，另一个是为了高级应用。它们提供了书写该类型 S-函数的模板，删

除注释后的具体内容如下。

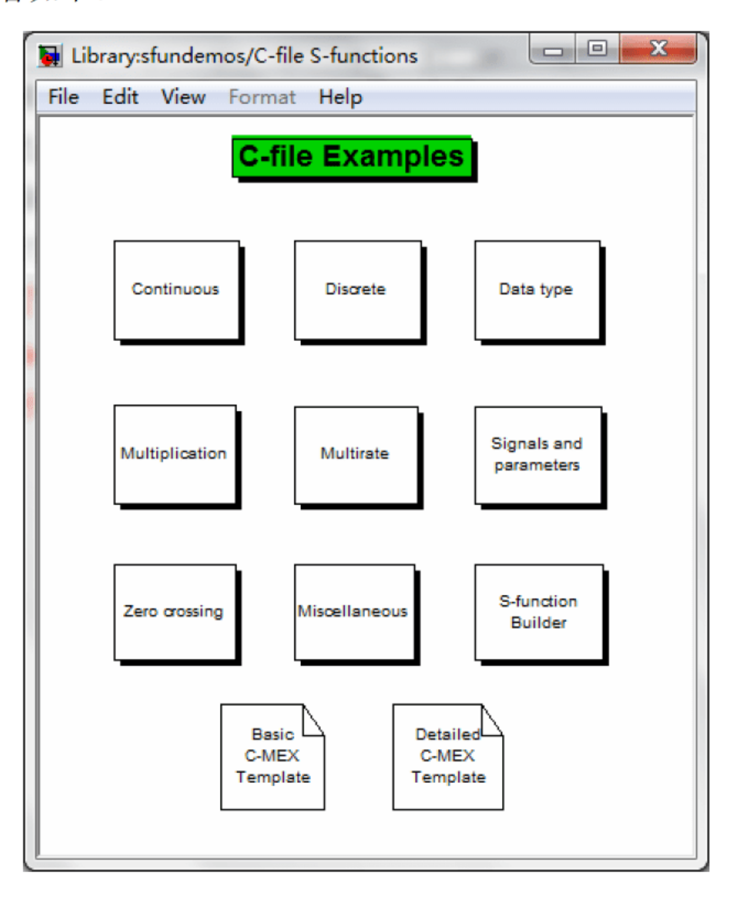


图 9-24 C-MEX 文件型 S-函数实例演示模块

1）Basic C-MEX Template

```
#define S_FUNCTION_NAME  sfuntmpl_basic
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
     /* Return if number of expected != number of actual parameters */
        return;
    }
    ssSetNumContStates(S, 0);
    ssSetNumDiscStates(S, 0);
    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortRequiredContiguous(S,0,true);/*direct  input signal access*/
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 1);
    ssSetNumSampleTimes(S, 1);
    ssSetNumRWork(S, 0);
    ssSetNumIWork(S, 0);
```

```
    ssSetNumPWork(S, 0);
    ssSetNumModes(S, 0);
    ssSetNumNonsampledZCs(S, 0);
    /* Specify the sim state compliance to be same as a built-in block */
    ssSetSimStateCompliance(S, USE DEFAULT SIM STATE);
    ssSetOptions(S, 0);
}
 /* Function: mdlInitializeSampleTimes*/
 static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS SAMPLE TIME);
    ssSetOffsetTime(S, 0, 0.0);
}
#define MDL_INITIALIZE_CONDITIONS  /* Change to #undef to remove function */
#if defined(MDL INITIALIZE CONDITIONS)
 static void mdlInitializeConditions(SimStruct *S)
  {
  }
#endif /* MDL_INITIALIZE_CONDITIONS */

#define MDL_START  /* Change to #undef to remove function */
#if defined(MDL START)
 static void mdlStart(SimStruct *S)
  {
  }
#endif /*  MDL_START */

static void mdlOutputs(SimStruct *S, int_T tid)
{
    const real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    real_T      *y = ssGetOutputPortSignal(S,0);
    y[0] = u[0];
}

#define MDL_UPDATE  /* Change to #undef to remove function */
#if defined(MDL UPDATE)
  static void mdlUpdate(SimStruct *S, int_T tid)
  {
  }
#endif /* MDL_UPDATE */

#define MDL_DERIVATIVES  /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)
  static void mdlDerivatives(SimStruct *S)
  {
  }
#endif /* MDL_DERIVATIVES */

static void mdlTerminate(SimStruct *S)
{
}
```

```
#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

2）Detailed C_MEX Template

```
#define S FUNCTION NAME  your sfunction name here
#define S FUNCTION LEVEL 2
#include "simstruc.h"
 #define MDL_CHECK_PARAMETERS   /* Change to #undef to remove function */
#if defined(MDL CHECK PARAMETERS) && defined(MATLAB MEX FILE)
    static void mdlCheckParameters(SimStruct *S)
  {
  }
#endif /* MDL_CHECK_PARAMETERS */
#define MDL_PROCESS_PARAMETERS   /* Change to #undef to remove function */
#if defined(MDL PROCESS PARAMETERS) && defined(MATLAB MEX FILE)
  static void mdlProcessParameters(SimStruct *S)
  {
  }
#endif /* MDL_PROCESS_PARAMETERS */

static void mdlInitializeSizes(SimStruct *S)
{
    int_T nInputPorts  = 1;  /* number of input ports  */
    int_T nOutputPorts = 1;  /* number of output ports */
    int_T needsInput   = 1;  /* direct feed through    */
    int_T inputPortIdx  = 0;
    int_T outputPortIdx = 0;
    ssSetNumSFcnParams(S, 0);  /* Number of expected parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
      return;
    }
    ssSetNumContStates(    S, 0);   /* number of continuous states  */
    ssSetNumDiscStates(    S, 0);   /* number of discrete states */
    if (!ssSetNumInputPorts(S, nInputPorts)) return;
    if(!ssSetInputPortDimensionInfo(S,inputPortIdx,DYNAMIC_DIMENSION)) return;
        ssSetInputPortDirectFeedThrough(S, inputPortIdx, needsInput);
     if (!ssSetNumOutputPorts(S, nOutputPorts)) return;
     if(!ssSetOutputPortDimensionInfo(S,outputPortIdx,DYNAMIC_DIMENSION)) return;
     ssSetNumSampleTimes(S, 1);   /* number of sample times                */
     ssSetNumRWork(S, 0);   /* number of real work vector elements    */
     ssSetNumIWork(S, 0);   /* number of integer work vector elements*/
     ssSetNumPWork(S, 0);   /* number of pointer work vector elements*/
     ssSetNumModes(S, 0);   /* number of mode work vector elements    */
     ssSetNumNonsampledZCs( S, 0);   /* number of nonsampled zero crossings   */
     ssSetSimStateCompliance(S, USE_DEFAULT_SIM_STATE);
     ssSetOptions(S, 0);   /* general options (SS_OPTION_xx)        */
 } /* end mdlInitializeSizes */
```

```
#define MDL_SET_INPUT_PORT_FRAME_DATA  /* Change to #undef to remove function */
#if defined(MDL_SET_INPUT_PORT_FRAME_DATA) && defined(MATLAB_MEX_FILE)
  static void mdlSetInputPortFrameData(SimStruct *S,  int  portIndex,
  Frame_T   frameData)
  {
  } /* end mdlSetInputPortFrameData */
#endif /* MDL_SET_INPUT_PORT_FRAME_DATA */

  #define MDL_SET_INPUT_PORT_WIDTH   /* Change to #undef to remove function */
#if defined(MDL_SET_INPUT_PORT_WIDTH) && defined(MATLAB_MEX_FILE)
   static void mdlSetInputPortWidth(SimStruct *S, int portIndex, int width)
  {
  } /* end mdlSetInputPortWidth */
#endif /* MDL_SET_INPUT_PORT_WIDTH */

 #define MDL_SET_OUTPUT_PORT_WIDTH   /* Change to #undef to remove function */
#if defined(MDL_SET_OUTPUT_PORT_WIDTH) && defined(MATLAB_MEX_FILE)
  static void mdlSetOutputPortWidth(SimStruct *S, int portIndex, int width)
  {
  } /* end mdlSetOutputPortWidth */
#endif /* MDL_SET_OUTPUT_PORT_WIDTH */

 #undef MDL_SET_INPUT_PORT_DIMENSION_INFO /* Change to #define to add function */
#if defined(MDL_SET_INPUT_PORT_DIMENSION_INFO) && defined(MATLAB_MEX_FILE)
    static void mdlSetInputPortDimensionInfo(SimStruct *S, int_T  portIndex,
const DimsInfo_T *dimsInfo)
  {
  } /* mdlSetInputPortDimensionInfo */
#endif /* MDL_SET_INPUT_PORT_DIMENSION_INFO */

 #undef MDL_SET_OUTPUT_PORT_DIMENSION_INFO /*Change to #define to add function*/
#if defined(MDL_SET_OUTPUT_PORT_DIMENSION_INFO) && defined(MATLAB_MEX_FILE)
   static void mdlSetOutputPortDimensionInfo(SimStruct  *S, int_T portIndex,
        const DimsInfo_T *dimsInfo)
  {
  } /* mdlSetOutputPortDimensionInfo */
#endif /* MDL_SET_OUTPUT_PORT_DIMENSION_INFO */

 #undef MDL_SET_DEFAULT_PORT_DIMENSION_INFO /* Change to #define to add fcn */
#if defined(MDL_SET_DEFAULT_PORT_DIMENSION_INFO) && defined(MATLAB_MEX_FILE)
  static void mdlSetDefaultPortDimensionInfo(SimStruct *S)
  {
  } /* mdlSetDefaultPortDimensionInfo */
#endif /* MDL_SET_DEFAULT_PORT_DIMENSION_INFO */

 #define MDL_SET_INPUT_PORT_SAMPLE_TIME
#if defined(MDL_SET_INPUT_PORT_SAMPLE_TIME) && defined(MATLAB_MEX_FILE)
 static void mdlSetInputPortSampleTime(SimStruct  *S, int_T  portIdx,
 real_T  sampleTime,
        real_T  offsetTime)
```

```
  {
  } /* end mdlSetInputPortSampleTime */
#endif /* MDL_SET_INPUT_PORT_SAMPLE_TIME */

 #define MDL SET OUTPUT PORT SAMPLE TIME
#if defined(MDL SET OUTPUT PORT SAMPLE TIME) && defined(MATLAB MEX FILE)
    static void mdlSetOutputPortSampleTime(SimStruct *S, int_T portIdx,
    real T sampleTime,
        real T offsetTime)
  {
  } /* end mdlSetOutputPortSampleTime */
#endif /* MDL_SET_OUTPUT_PORT_SAMPLE_TIME */

static void mdlInitializeSampleTimes(SimStruct *S)
{
    /* Register one pair for each sample time */
    ssSetSampleTime(S, 0, CONTINUOUS SAMPLE TIME);
    ssSetOffsetTime(S, 0, 0.0);
 } /* end mdlInitializeSampleTimes */
 #define MDL_SET_INPUT_PORT_DATA_TYPE  /* Change to #undef to remove function */
#if defined(MDL SET INPUT PORT DATA TYPE) && defined(MATLAB MEX FILE)
    static void mdlSetInputPortDataType(SimStruct *S, int portIndex,DTypeId
    dType)
  {
  } /* mdlSetInputPortDataType */
#endif /* MDL_SET_INPUT_PORT_DATA_TYPE */

  #define MDL_SET_OUTPUT_PORT_DATA_TYPE /* Change to #undef to remove function */
#if defined(MDL SET OUTPUT PORT DATA TYPE) && defined(MATLAB MEX FILE)
   static void mdlSetOutputPortDataType(SimStruct *S,int portIndex,DTypeId dType)
  {
  } /* mdlSetOutputPortDataType */
#endif /* MDL_SET_OUTPUT_PORT_DATA_TYPE */

 #define MDL_SET_DEFAULT_PORT_DATA_TYPES /* Change to #undef to remove function*/
#if defined(MDL SET DEFAULT PORT DATA TYPES) && defined(MATLAB MEX FILE)
   static void mdlSetDefaultPortDataTypes(SimStruct *S)
  {
  } /* mdlSetDefaultPortDataTypes */
#endif /* MDL_SET_DEFAULT_PORT_DATA_TYPES */

 #define MDL_SET_INPUT_PORT_COMPLEX_SIGNAL  /* Change to #undef to remove */
#if defined(MDL_SET_INPUT_PORT_COMPLEX_SIGNAL) && defined(MATLAB_MEX_FILE)
  static void mdlSetInputPortComplexSignal(SimStruct  *S,
        int       portIndex,
        CSignal_T cSignalSetting)
  {
  } /* mdlSetInputPortComplexSignal */
#endif /* MDL_SET_INPUT_PORT_COMPLEX_SIGNAL */

 #define MDL_SET_OUTPUT_PORT_COMPLEX_SIGNAL  /* Change to #undef to remove */
```

```
#if defined(MDL SET OUTPUT PORT COMPLEX SIGNAL) && defined(MATLAB MEX FILE)
   static void mdlSetOutputPortComplexSignal(SimStruct  *S,
        int       portIndex,
        CSignal T cSignalSetting)
  {
  } /* mdlSetOutputPortComplexSignal */
#endif /* MDL_SET_OUTPUT_PORT_COMPLEX_SIGNAL */

  #define MDL_SET_DEFAULT_PORT_COMPLEX_SIGNALS /* Change to #undef to remove */
#if
defined(MDL SET DEFAULT PORT COMPLEX SIGNALS)&& defined(MATLAB MEX FILE)
  static void mdlSetDefaultPortComplexSignals(SimStruct *S)
  {
  } /* mdlSetDefaultPortComplexSignals */
#endif /* MDL_SET_DEFAULT_PORT_COMPLEX_SIGNALS */

#define MDL_SET_WORK_WIDTHS   /* Change to #undef to remove function */
#if defined(MDL SET WORK WIDTHS) && defined(MATLAB MEX FILE)
    static void mdlSetWorkWidths(SimStruct *S)
  {
  }
#endif /* MDL_SET_WORK_WIDTHS */

#define MDL_INITIALIZE_CONDITIONS   /* Change to #undef to remove function */
#if defined(MDL INITIALIZE CONDITIONS)
    static void mdlInitializeConditions(SimStruct *S)
  {
  }
#endif /* MDL_INITIALIZE_CONDITIONS */

 #define MDL_START  /* Change to #undef to remove function */
#if defined(MDL START)
    static void mdlStart(SimStruct *S)
  {
  }
#endif /*  MDL_START */

#define MDL_SIM_STATE /* Change to #undef to remove this function */
#if defined(MDL SIM STATE)
    static mxArray* mdlGetSimState(SimStruct* S)
  {
  }
  static void mdlSetSimState(SimStruct* S, const mxArray* inSimState)
  {
  }
#endif /* MDL_SIM_STATE */

#define MDL_GET_TIME_OF_NEXT_VAR_HIT  /* Change to #undef to remove function */
#if defined(MDL_GET_TIME_OF_NEXT_VAR_HIT) && (defined(MATLAB_MEX_FILE) || \
                                               defined(NRT))
    static void mdlGetTimeOfNextVarHit(SimStruct *S)
```

```
  {
      time_T timeOfNextHit = ssGetT(S) /* + offset */ ;
      ssSetTNext(S, timeOfNextHit);
  }
#endif /* MDL_GET_TIME_OF_NEXT_VAR_HIT */

#define MDL_ZERO_CROSSINGS   /* Change to #undef to remove function */
#if defined(MDL_ZERO_CROSSINGS) && (defined(MATLAB_MEX_FILE) || defined(NRT))

  static void mdlZeroCrossings(SimStruct *S)
  {
  }
#endif /* MDL_ZERO_CROSSINGS */

static void mdlOutputs(SimStruct *S, int_T tid)
{
} /* end mdlOutputs */

#define MDL_UPDATE  /* Change to #undef to remove function */
#if defined(MDL_UPDATE)

  static void mdlUpdate(SimStruct *S, int_T tid)
  {
  }
#endif /* MDL_UPDATE */

#define MDL_DERIVATIVES  /* Change to #undef to remove function */
#if defined(MDL_DERIVATIVES)

  static void mdlDerivatives(SimStruct *S)
  {
  }
#endif /* MDL_DERIVATIVES */

static void mdlTerminate(SimStruct *S)
{
}

#define MDL_RTW  /* Change to #undef to remove function */
#if defined(MDL_RTW) && defined(MATLAB_MEX_FILE)
  static void mdlRTW(SimStruct *S)
  {
  }
#endif /* MDL_RTW */
#ifdef  MATLAB_MEX_FILE    /* Is this file being compiled as a MEX-file? */
#include "simulink.c"      /* MEX-file interface mechanism */
#else
#include "cg_sfun.h"       /* Code generation registration function */
#endif
```

9.5.2 编写方法

本节将对 Basic C-MEX 模板中的各部分加以详细说明。编写 S-函数就是根据需求，用相应的代码去代替模板中对应部分的代码。

1．Level-2 M 文件型和 C MEX 文件型 S-函数的关系

Level-2 文件型与 C MEX 文件型 S-函数在书写上比较类似，表 9-11 中列出了两者在回调函数上的等价关系。

表 9-11 Level-2 M 文件型与 C MEX 文件型 S-函数回调函数的等价关系

Level-2 M 文件型	C MEX 文件型
setup	mdlInitializeSizes
CheckParameters	mdlCheckParameters
Derivatives	mdlDerivatives
Disable	mdlDisable
Enable	mdlEnable
InitializeCondition	mdlInitializeConditions
Outputs	mdlOutputs
PostPropagationSetup	mdlSetWorkWidths
ProcessParameters	mdlProcessParameters
Projection	mdlProjection
SetInputPortComplexSignal	mdlSetInputPortComplexSignal
SetInputPortDataType	mdlSetInputPortDataType
SetInputPortDimensions	mdlSetInputPortDimensionInfo
SetInputPortSampleTime	mdlSetInputPortSampleTime
SetInputPortSampleMode	mdlSetInputPortFrameData
SetOutputPortComplexSignal	mdlSetOutputPortComplexSignal
SetOutputPortComplexSignal	mdlSetOutputPortComplexSignal
SetOutputPortDataType	mdlSetOutputPortDataType
SetOutputPortDimesions	mdlSetOutputPortDimensionInfo
SetOutputPortSampleTime	mdlSetOutputPortSampleTime
SimStatusChange	mdlSimStatusChange
Start	mdlStart
Terminate	mmdlTerminate
Level-2 M 文件型	C MEX 文件型
Update	mdlUpdate
WriteRTW	mdlRTW

2．C MEX 文件型 S-函数的特性

C MEX 文件型 S-函数的编写满足 C 语言的规范，这是与前面 M 文件型 S-函数的最大区别，下面具体解释各段代码的含义。

1）S-函数声明

```
#define S_FUNCTION_NAME  MySfunction  /*函数命名*/
```

```
#define S_FUNCTION_LEVEL 2  /*指定 LEVEL 2 类型*/
```

2）导入支持 SimStruct 的库文件

```
#include "simstruc.h"
```

3）模块初始化回调函数

```
static void mdlInitializeSizes(SimStruct *S)
                              /*S 相当于 Level-2 M 文件型 S-函数中的 block*/
{
ssSetNumSFcnParams(S,0);        /*设置外部参数个数*/
if(ssGetNumSFcnParams(S)!=ssGetSFcnParamsCount(S))
{
    return;                     /*期望参数个数不等于实际参数个数的处理方法*/
}
ssSetNumContStates(S,0)         /*设置连续状态个数*/
ssSetNumDiscStates(S,0)         /*设置离散状态个数*/
if(!ssSetNumInputPorts(S,1))
   return;  /*ssSetNumInputPorts(S,1)设置输入端口个数为 1*/
ssSetInputPortWidth(S,0,1);
/*设置第一个输入端口的输入个数，C 语言中第一个元素下标为 0*/
ssSetInputPortRequiredContinuous(S,0,true); /*设置第一个输入端口始终输入*/
ssSetInputPortDirectFeedThrough(S,0,1);     /*设置第一个输入端口有直接馈入*/
if(!ssSetNumOutputPorts(S,1))
   return;  /*ssSetNumOutputPorts(S,1)设置输出端口个数为 1*/
ssSetOutputPortWidth(S,0,1);    /*设置第一个输出端口的输入个数*/
ssSetNumSampleTimes(S,1);       /*设置采样时间个数为 1*/
ssSetNumRWork(S,0);             /*设置实数元素个数*/
ssSetNumIWork(S,0);             /*设置整数元素个数*/
ssSetNumPWork(S,0);             /*设置指针元素个数*/
ssSetNumMWork(S,0);             /*设置模式元素个数*/
ssSetNumNonsampledZCs(S,0);     /*设置非采样过零个数*/
ssSetOptions(S,0);              /*具体选项设置在 simstruc.h 文件中*/
}
```

4）采样时间初始化回调函数

```
static void mdlIntializeSampleTimes(SimStruct *S)
{
ssSetSampleTime(S,0,CONTINUOUS_SAMPLE_TIME); /*为第一个采样时间设置采样周期*/
ssSetOffsetTime(S,0,0.0);  /*为第一个采样时间设置偏置量*/
}
```

5）状态初始值设置回调函数

```
#define MDL INITIALIZE CONDITIONS
#if defined(MDL INITIALIZE CONDITIONS)
   static void mdlIntializeConditions(SimStruct *S)
   {
      real_T  *xc0 = ssGetContStates(S);/*设定连续状态初始值的指针，进而赋值*/
```

```
        real_T  *xd0 = ssGetRealDiscStates(S);
/*设定离散状态初始值的指针，进而赋值*/
}
```

6）模块自动启动设置回调函数

```
#define MDL START
#if defined(MDL START)
static void mdlStart(SimStruct *S)
{
}
#endif
```

7）输出回调函数

```
static void mdlOutputs(SimStruct *S,int_T tid)
{
const real_T *u = (const real_T*)ssGetInputPortSignal(S,0);
 /*将第 1 个输入端口的值赋给变量*/
real_T *y = ssGetOutputPortSignal(S,0);
/*声明第 1 个输出端口值所对应的指针*/
y[0] = u[0];  /*将待输出的值赋给第 1 个输出端口*/
}
```

8）更新回调函数

```
#define MDL UPDATE
#if defined(MDL UPDATE)
static void mdlUpdate(SimStruct *S,int_T tid)
{
*real_T *x = ssGetRealDiscStates(S);  /*将连续状态的值赋给指定变量*/
*const real_T *u = (const real_T*)ssGetInputPortSignal(S,0);
/*将第 1 个输入端口的值赋给变量*/
}
#endif
```

9）微分回调函数

```
#define MDL_DERIVATIVES
#if defined(MDL_DERIVATES)
{
*real_T  *dx = ssGetdX(S);  /*将连续状态的导数值赋给指定变量*/
*real_T  *x = ssGetContStates(S);  /*将连续状态的值赋给指定变量*/
*const real_T  *u = (const real_T*)ssGetInputPortSignal(S,0);
/*将第 1 个输入端口的值赋给变量*/
}
#endif
```

10）终止回调函数

```
static void mdlTermiante(SimStruct *S)
{
}
```

11）S-函数结尾

```
#ifdef MATLAB_MEX_FILE
#include "simulink.c"  /*加载 simulink.c*/
#else
#include "cg_sfun.h"  /*加载 simulink.cg_sfun.h*/
#endif
```

综上，详细介绍了 C MEX 文件型 S-函数模板的内容，这为 C MEX 文件型 S-函数的书写提供了便利。

9.5.3 实例

前面介绍了 C MEX 文件型 S-函数的写法，下面通过不同类型的实例进一步说明 C MEX 文件型 S-函数的编写方法和编写技巧。

1．连续系统

下面给出一个连续系统的例子。

【例 9-9】 用 C MEX 文件型 S-函数（MySFunction9.c）描述方程 $\begin{cases} \dot{X} = AX + Bu \\ X(0) = 0 \\ Y = CX + Du \end{cases}$ （其中 $A = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -1 & -2 & -3 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, C = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, D = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$），其内容如下。

```
#define S_FUNCTION_NAME MySfunction9
#define S_FUNCTION_LEVEL 2
#include "simstruc.h"
 static void mdlInitializeSizes(SimStruct *S)
{
ssSetNumContStates(S, 3);
 if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortRequiredContiguous(S, 0, true);
    ssSetInputPortDirectFeedThrough(S, 0, 1);
    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 2);
    ssSetNumSampleTimes(S, 1);
    ssSetOptions(S, 0);
}
 static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, CONTINUOUS_SAMPLE_TIME);
    ssSetOffsetTime(S, 0, 0.0);
}

#define MDL_INITIALIZE_CONDITIONS
static void mdlInitializeConditions(SimStruct *S)
{
    real_T  *x0 = ssGetContStates(S);
    int_T  i;
    //X0=0
```

```
    for (i=0; i<3; i++) {
        x0[i] = 0;
    }
}
 static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T *y = ssGetOutputPortRealSignal(S,0);
    real_T *x = ssGetContStates(S);
    real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    //Y=CX+Du
    y[0] = x[0] + u[0];
    y[1] = x[1] + 2*u[0];
}

#define MDL DERIVATIVES
static void mdlDerivatives(SimStruct *S)
{
    real_T*dx= ssGetdX(S);
    real_T*x= ssGetContStates(S);
    real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    //dX=AX+Bu
    dx[0] =  x[1];
    dx[1] =  x[2];
    dx[2] =  -2*x[0]-3*x[1]-5*x[2]+u[0];
}

static void mdlTerminate(SimStruct *S)
{
 }

#ifdef  MATLAB MEX FILE
#include "simulink.c"
#else
#include "cg sfun.h"
#endif
```

在仿真前，需要首先使用如下指令编译 MySfunction9.c 文件：

```
mex MySfunction9.c
```

此时，MySfunction9.c 文件所在目录下增加了一个文件 MySfunction9.mexw32。

在 Simulink 中，使用该 S-函数的实例如图 9-25 所示，运行后双击 Scope 模块可得到如图 9-26 所示的结果。

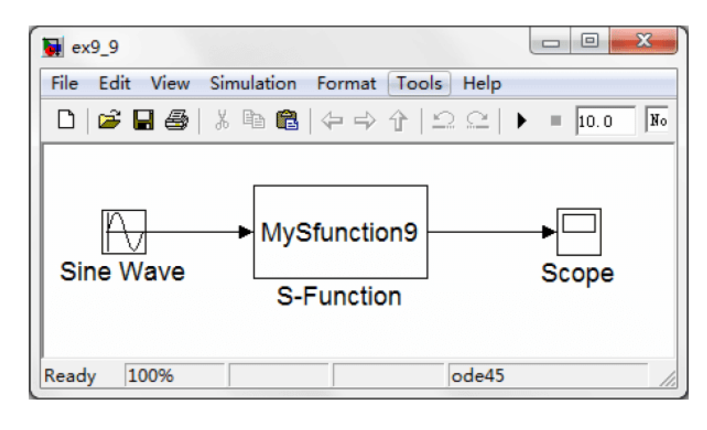


图 9-25　Simulink 仿真框图

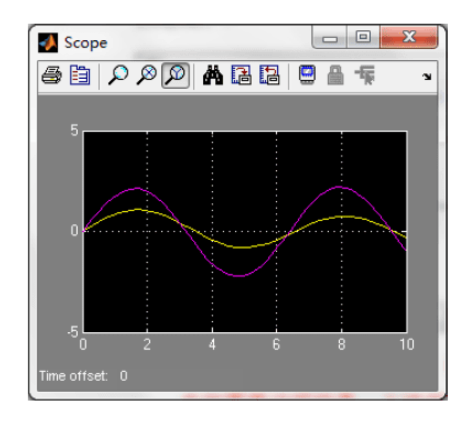


图 9-26　Simulink 仿真运行结果

本例需要说明的是，上述代码不支持中文注释。

2. 离散系统

下面给出一个离散系统的例子，同时将外部参数传入 S-函数。

【例 9-10】 用 C MEX 文件型 S-函数（MySfunction10.c）描述方程 $\begin{cases} X(n+1)=AX(n)+Bu(n) \\ X(0)=0 \\ Y(n)=CX(n)+Du(n) \end{cases}$（其中 $A=\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ d1 & d2 & d3 \end{bmatrix}, B=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, C=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, D=\begin{bmatrix} 1 \\ q \end{bmatrix}$, $E=(d1 \quad d2 \quad d3)$），其内容如下。

```
#define S_FUNCTION_NAME MySfunction10
#define S_FUNCTION_LEVEL 2

#include "simstruc.h"

static void mdlInitializeSizes(SimStruct *S)
{
    ssSetNumSFcnParams(S, 2);  /* 2 parameters */
    if (ssGetNumSFcnParams(S) != ssGetSFcnParamsCount(S)) {
       return;
    }

    ssSetNumDiscStates(S, 3);

    if (!ssSetNumInputPorts(S, 1)) return;
    ssSetInputPortWidth(S, 0, 1);
    ssSetInputPortRequiredContiguous(S, 0, true);
    ssSetInputPortDirectFeedThrough(S, 0, 1);

    if (!ssSetNumOutputPorts(S, 1)) return;
    ssSetOutputPortWidth(S, 0, 2);

    ssSetNumSampleTimes(S, 1);

    ssSetOptions(S, 0);
}

static void mdlInitializeSampleTimes(SimStruct *S)
{
    ssSetSampleTime(S, 0, 0.1);
    ssSetOffsetTime(S, 0, 0.0);
 }

#define MDL_INITIALIZE_CONDITIONS   /* Change to #undef to remove function */
#if defined(MDL_INITIALIZE_CONDITIONS)
  static void mdlInitializeConditions(SimStruct *S)
  {
   real_T *x0 = ssGetRealDiscStates (S);
    int_T  i;
```

```
    //X0=0
    for (i=0; i<3; i++) {
        x0[i] = 0;
    }
  }
#endif /* MDL_INITIALIZE_CONDITIONS */

static void mdlOutputs(SimStruct *S, int_T tid)
{
    real_T  *y = ssGetOutputPortRealSignal(S,0);
    real_T  *x = ssGetRealDiscStates (S);
    real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    real_T *p2= mxGetPr(ssGetSFcnParam(S,1));
    //Y=CX+Du
    y[0] = x[0] + u[0];
    y[1] = x[1] + p2[0]* u[0];
}

#define MDL_UPDATE  /* Change to #undef to remove function */
#if defined(MDL UPDATE)
   static void mdlUpdate(SimStruct *S, int_T tid)
  {
    real_T  *y= ssGetOutputPortRealSignal(S,0);
    real_T  *x= ssGetRealDiscStates (S);
    real_T *u = (const real_T*) ssGetInputPortSignal(S,0);
    real_T  *p1= mxGetPr(ssGetSFcnParam(S,0));
    real T  tmpx[3]={ 0,0,0};
    //X(k+1)=AX(k)+Bu(k)
    tmpx[0]=-x[1];
    tmpx[1]=-x[2];
    tmpx[2]=p1[0]*x[0]+p1[1]*x[1]+ p1[2]*x[2]+u[0];
    x[0]=tmpx[0];
x[1]=tmpx[1];
x[2]=tmpx[2];
  }
#endif /* MDL_UPDATE */

static void mdlTerminate(SimStruct *S)
{
}

#ifdef  MATLAB_MEX_FILE
#include "simulink.c"
#else
#include "cg_sfun.h"
#endif
```

在仿真前，需要首先使用如下指令编译 MySfunction10.c 文件：

```
mex MySfunction10.c
```

在 Simulink 中，使用该 S-函数的实例如图 9-27 所示。当 D=($d1$ $d2$ $d3$)和 $q = 2$ 时，双

击 S-Function 模块，如图 9-28 所示设置 S-function parameters 属性即可，运行结果如图 9-29 所示。

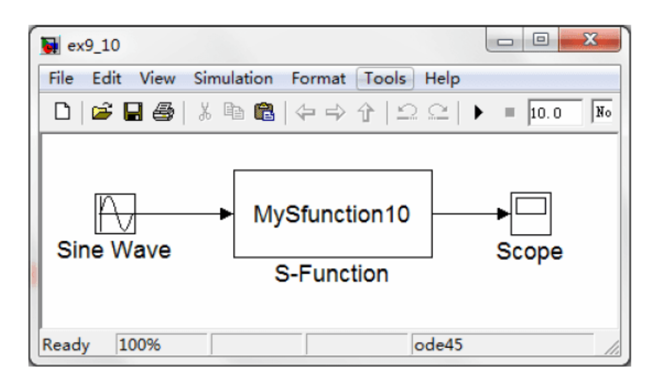


图 9-27　Simulink 仿真框图

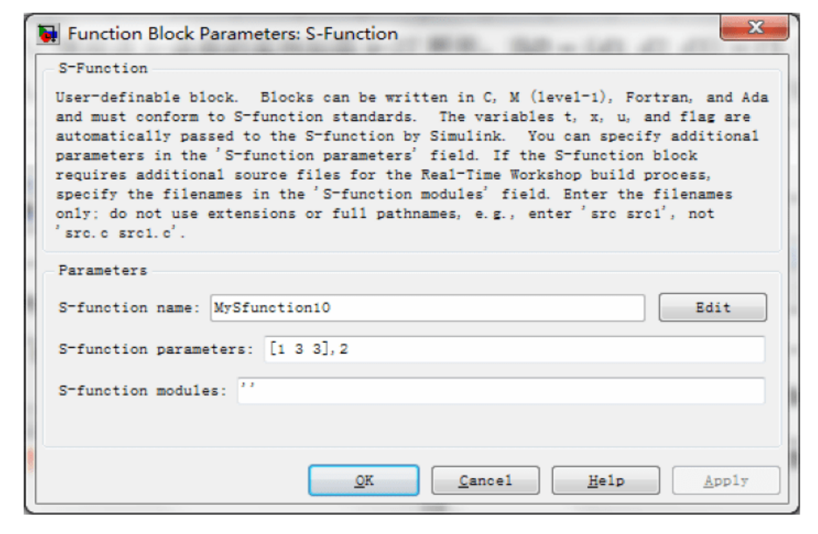


图 9-28　S-函数参数设置

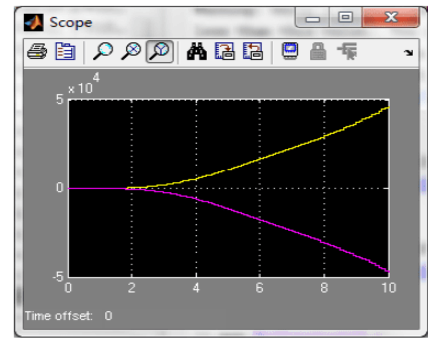


图 9-29　Simulink 仿真运行结果

若将函数声明和回调函数 mdlUpdate 分别做如下替换并保存为 MySfunction102.c，编译和设置参数后运行结果如图 9-22 所示。

```
#define S FUNCTION NAME MySFunction102
static void mdlUpdate(SimStruct *S,int_T tid)
{
   real_T *y = ssGetOutputPortRealSignal(S,0);
   real_T *x = ssGetRealDiscStates(S);
   real_T *u = (const real_T*)ssGetInputPortSignal(S,0);
   real_T *p1=mxGetPr(ssGetSFcnParam(S,0));
   x[0] = x[1];
   x[1] = x[2];
   x[2] = p1[0]*x[0]+p1[1]*x[1]+p1[2]*x[2]+u[0];
}
```

这里需要注意的是，本例对于相同的方程却有两个不同的结果。S-函数 MySfunction10.c 和 MySfunction5.m 采用的是同步解差分方程的方式，即在计算新状态值时是同步更新的；而 S-函数 MySfunction102.c 和 MySfunction8.m 采用的是异步计算差分方

程的方式，即在计算新状态值时按照指定次序更新，且已更新状态的结果用于后面状态的更新；对于异步方式，通过改变更新次序，也将获得不同的仿真结果。

对于连续系统而言，不存在上述的差异。

9.6 使用 S-函数创建器编写 C MEX 文件型

前面介绍了如何使用各类模板生成 S-函数，MATLAB 还提供了 S-函数创建器（S-functions Builder），以便通过界面生成 C MEX 文件型 S-函数。

下面通过一个实例说明基于 S-函数创建器生成 C MEX 文件型 S-函数的方法。

【例 9-11】 用 S-函数创建器生成 S-函数（MySfunction11.c）描述方程 $\begin{cases} X(n+1)=AX(n)+Bu(n) \\ X(0)=0 \\ Y(n)=CX(n)+Du(n) \end{cases}$（其中 $A=\begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & -1 \\ p1 & p2 & p3 \end{bmatrix}, B=\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, C=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}, D=\begin{bmatrix} 1 \\ q \end{bmatrix},$ $E=(p1 \quad p2 \quad p3)$）。具体步骤如下。

（1）创建如图 9-30 所示的 Simulink 仿真初始框图。

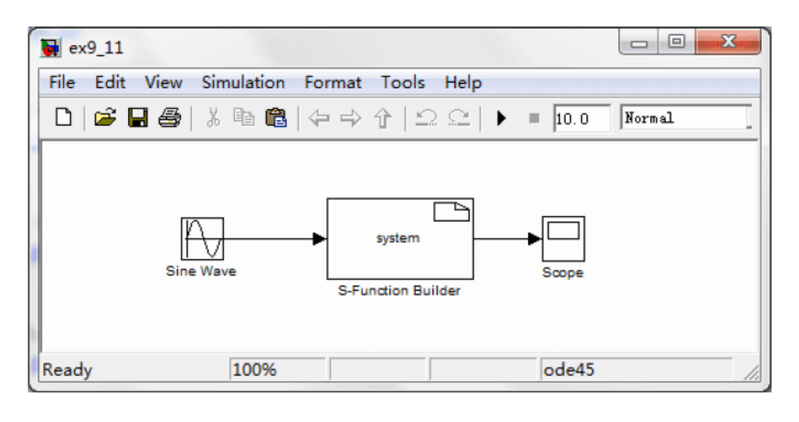


图 9-30　Simulink 仿真初始框图

（2）双击“system”模块，可以看到如图 9-31 所示的 S-函数设计界面。

（3）在如图 9-31 的 S-函数设计界面上填写如下内容。

- 在“S-function name（文件名）”处填写“MySfunction11”。
- 在“Initialization（初始化）”选项卡的“Number of discrete states（离散状态个数）”处填写“3”，“Discrete states IC（初始状态值）”处填写“[0 0 0]”，“Sample mode（采样时间类型）”处选择“Discrete”，“Sample time value（采样时间值）”处填写“0.1”，目前不支持多采样时间和偏置量的设置。
- 在“Data Properties（数据属性）”选项卡的“Output ports”子选项卡下，选择 y0（端口 1）的“Rows（输出个数）”为 2。
- 在“Data Properties（数据属性）”选项卡的“Parameters”子选项卡下，先单击图标（增加参数），再将新增参数的“Parameter name（参数名称）”设置为 p1。
- 在“Data Properties（数据属性）”选项卡的“Parameters”子选项卡下，先单击图

标（增加参数），再将新增参数的“Parameter name（参数名称）”设置为 p2。

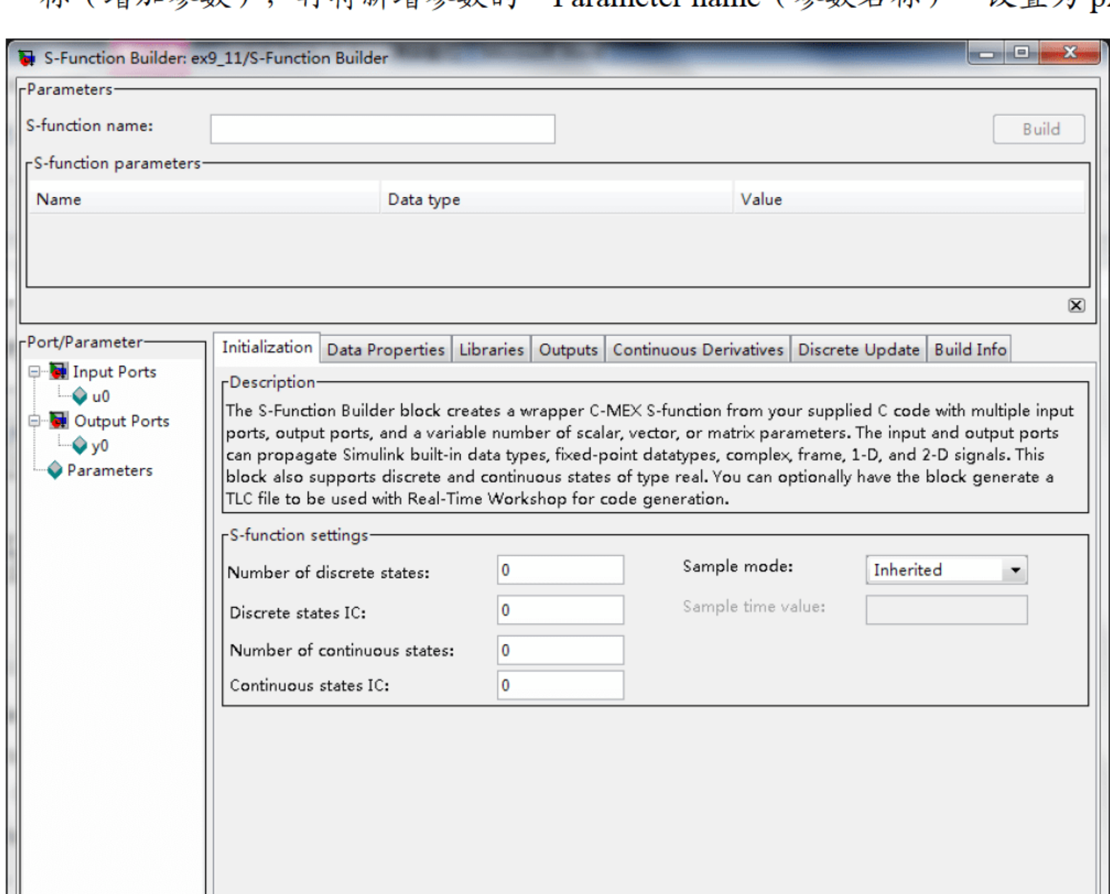


图 9-31　S-函数设计界面

❑ 在“Outputs（输出回调函数）”选项卡中填写如下内容。

```
y0[0] = xD[0]+u0[0];
y0[1] = xD[1]+p2[0]*u0[0];
```

其中，xD[0]表示第一个离散状态，u0[0]表示第一个输入端口的第一个输入，y0[0]表示第一个输出端口的第一个输出，p2[0]表示前面设置的参数 p2。同时，用 xC[0]表示第一个连续状态，dx[0]表示第一个连续状态的导数，若参数 P 是 m×n 的矩阵，则 P(i, j)应表示成 P[i+(j–i)*m]，即将数据按列拉直后排号。

❑ 在“S-function parameters（参数列表）”下参数 p1 的“Value（值）”处填写“[1　3　3]”，参数 p2 的“Value（值）”处填写“2”。

❑ 在“Discrete Update（离散状态更新回调函数）”选项卡中填写如下内容。

```
xD[0] = -xD[1];
xD[1] = -xD[2];
xD[2] = p1[0]*xD[0]+p1[1]*xD[1]+p1[2]*xD[2]+u0[0];
```

（4）单击主界面上的“Build”按钮，则自动生成文件 MySfunction11.c、MySfunction11_wrapper.c、MySfunction11.tlc 和 MySfunction11.mex32，得到如图 9-32 所示的编译结果。

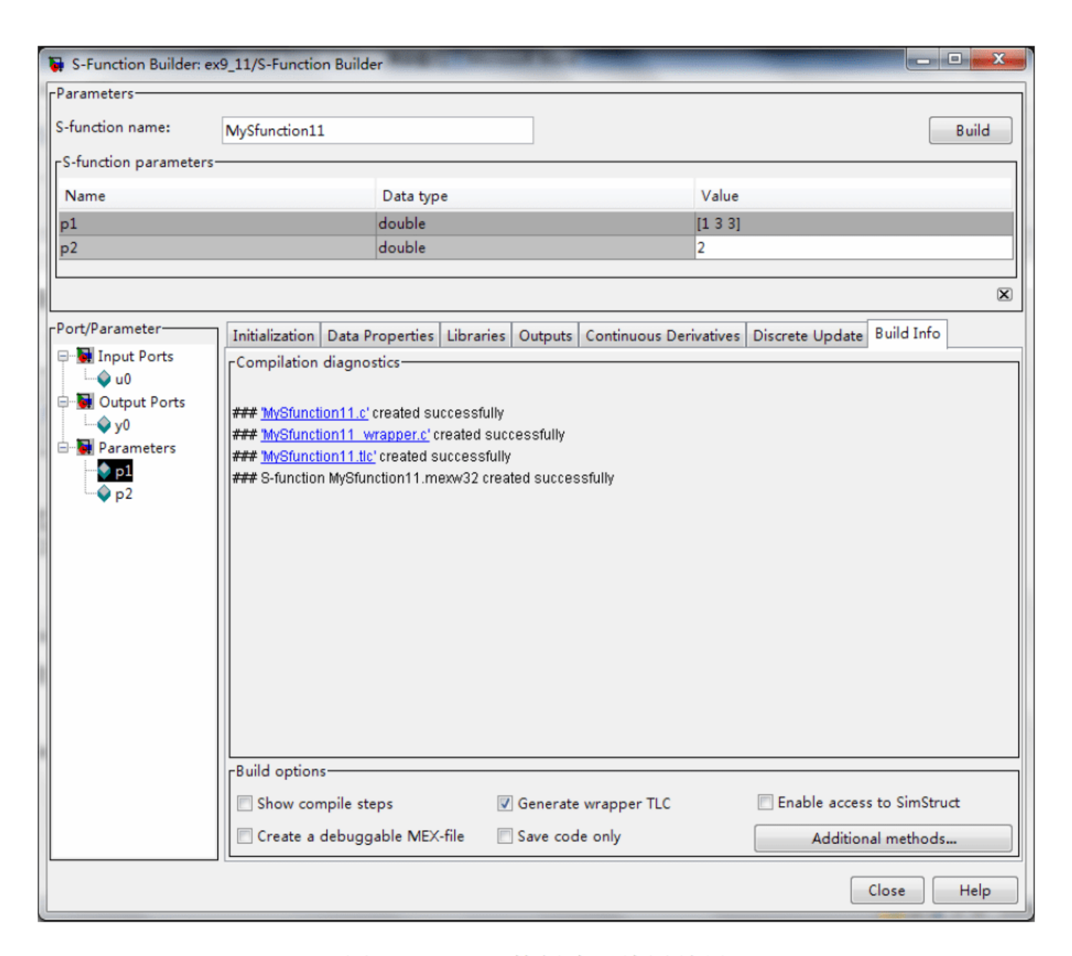


图 9-32　S-函数创建器编译结果

最后得到如图 9-33 所示的 Simulink 仿真框图，运行结果如图 9-34 所示。

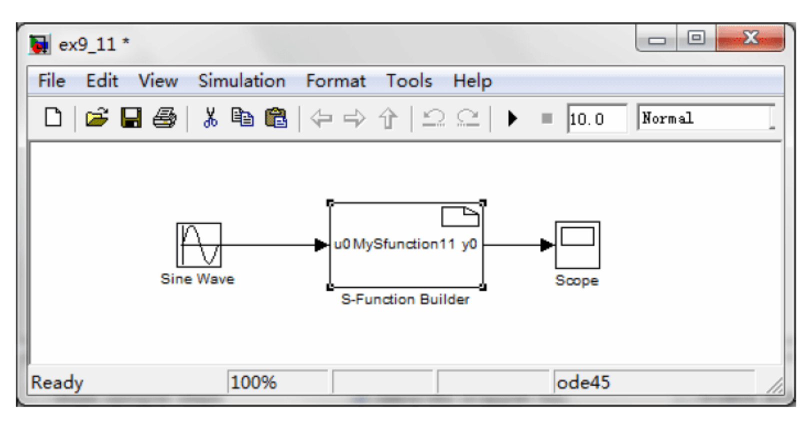


图 9-33　Simulink 仿真框图

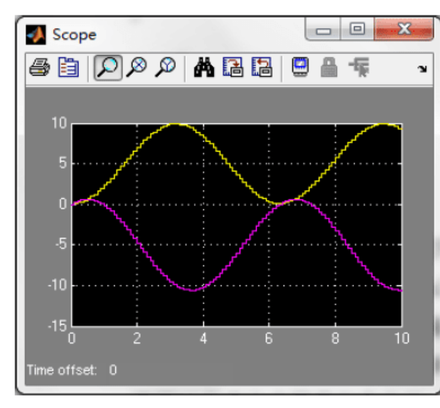


图 9-34　Simulink 运行结果

通过 S-函数创建器生成 C MEX 文件型 S-函数比较方便，但功能受到一定的限制，可根据实际情况选择生成的方法。

9.7　本章小结

本章主要介绍系统函数，即 S-函数。当用 MATLAB 所提供的模型不能完全满足用户，

则需要提供给用户自己编写程序来满足自己要求模型的接口。S-函数作为与其他语言相结合的接口，可以使用这个语言所提供的强大能力。S-Function 可以使用 MATLAB、C、C++、Ada 或 Fortran 语言来编写。本章分别介绍了使用各类模板生成 S-函数，重点介绍了 C MEX 文件型 S-函数的编写方法。

9.8 习题

（1）用 Level-1 M 文件型 S-函数描述方程 $\begin{cases}\dot{X}=AX+Bu\\X(0)=0\\Y(n)=CX+Du\end{cases}$（其中 $A=\begin{bmatrix}1&0&0\\0&1&0\\1&-1&4\end{bmatrix},B=\begin{bmatrix}0\\0\\1\end{bmatrix},C=\begin{bmatrix}2&1&0\\0&1&0\end{bmatrix},D=\begin{bmatrix}1\\1\end{bmatrix}$）。

（2）用 Level-2 M 文件型 S-函数描述方程 $\begin{cases}\dot{X}=AX+Bu\\X(0)=0\\Y(n)=CX+Du\end{cases}$（其中 $A=\begin{bmatrix}0&1&0\\0&0&1\\2&3&5\end{bmatrix},B=\begin{bmatrix}0\\0\\1\end{bmatrix},C=\begin{bmatrix}1&1&0\\0&1&0\end{bmatrix},D=\begin{bmatrix}1\\2\end{bmatrix}$）。

（3）用 C MEX 文件型 S-函数描述方程 $\begin{cases}\dot{X}=AX+Bu\\X(0)=0\\Y(n)=CX+Du\end{cases}$（其中 $A=\begin{bmatrix}0&1&0\\0&0&1\\1&2&3\end{bmatrix},B=\begin{bmatrix}0\\0\\1\end{bmatrix},C=\begin{bmatrix}1&1&0\\0&1&0\end{bmatrix},D=\begin{bmatrix}1\\2\end{bmatrix}$）。

第 10 章　MATLAB 工具箱

所谓 MATLAB 工具箱就是一些 M 文件的集合，用户可以修改工具箱中的函数，更为重要的是用户可以通过编制 M 文件来任意添加工具箱中原来没有的工具函数，此功能充分体现了 MATLAB 语言的开放性。MATLAB 拥有几十个功能强大的工具箱，每一个工具箱都是为某一学科和专业应用而特别指定的，这是 MATLAB 语言能够快速发展的重要因素之一。

10.1　MATLAB 工具箱简介

MATLAB 的工具箱大致可分为两类：一类是功能性工具箱，也就是通用型；另一类是领域性工具箱，也就是专用型。功能性工具箱主要用来扩充 MATLAB 的符号计算功能、图形建模仿真功能、文字处理功能及与硬件实时交互功能，能够用于多种学科；领域性工具箱是学科专用工具箱，适用于相关专业，如控制系统、信号处理、模糊逻辑以及金融工具箱等。

通过单击【Start】/【Toolboxs】，弹出如图 10-1 所示的 MATLAB 工具箱。

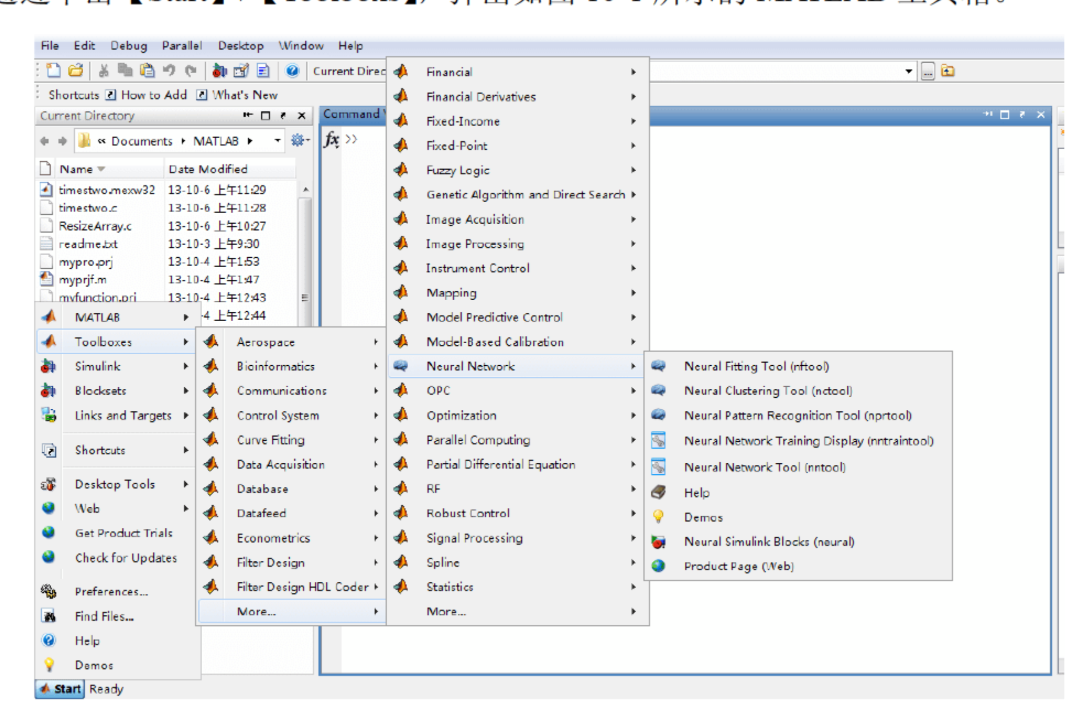


图 10-1　Matlab 工具箱

下面列出了一些 MATLAB 工具箱。

- ❑ Bioinformatics Toolbox——生物工具箱。
- ❑ Control System Toolbox——控制系统工具箱。
- ❑ Curve Fitting Toolbox——曲线拟合工具箱。

- ❑ Communications Toolbox——通信工具箱。
- ❑ Database Toolbox——数据库工具箱。
- ❑ Fuzzy Logic Toolbox——模糊逻辑工具箱。
- ❑ Filter Design Toolbox——滤波器设计工具箱。
- ❑ Financial Toolbox——金融工具箱。
- ❑ Fixed-Income Toolbox——固定收入工具箱。
- ❑ Fixed-Point Toolbox——定点运算工具箱。
- ❑ Financial Derivatives Toolbox——金融误差分析工具箱。
- ❑ Instrument Control Toolbox——器具控制工具箱。
- ❑ Image Processing Toolbox——图像处理工具箱。
- ❑ Image Acquisition Toolbox——图像采集工具箱。
- ❑ Mapping Toolbox——地图绘制工具箱。
- ❑ Model Predictive Control Toolbox——模型预测控制工具箱。
- ❑ Model-Based Calibration Toolbox——基于模型标准工具箱。
- ❑ Neural Network Toolbox——神经网络工具箱。
- ❑ Optimization Toolbox——最优化工具箱。
- ❑ Partial Differential Equation Toolbox——偏微分方程工具箱。
- ❑ Robust Control Toolbox——鲁棒控制工具箱。
- ❑ System Identification Toolbox——系统辨识工具箱。
- ❑ Signal Processing Toolbox——信号处理工具箱。
- ❑ Spline Toolbox——样条工具箱。
- ❑ Statistics Toolbox——统计学工具箱。
- ❑ Symbolic Math Toolbox——符号数学工具箱。
- ❑ Wavelet Toolbox——小波工具箱。

由于篇幅所限，不可能对所有工具箱的使用进行一一介绍。本章将对两个具有代表性的工具箱结构和使用方法进行介绍。读者可根据专业需要，参考帮助信息对不同工具箱的结构和使用方法进行学习。

10.2 神经网络工具箱

虽然神经网络有着广泛的实用性和强大的解决问题的能力，但是它也存在一些缺陷。比如，神经网络的建立实际上就是一个不断尝试的过程，以 BP 网络为例，网络的层次及每一层节点的个数都是通过不断训练进行改进的。同样，对于神经网络的学习过程来说，已经有很多成型的学习算法，但这些算法在数学计算上都比较复杂，过程也比较繁琐，容易出错。因此，采用计算机辅助进行神经网络设计与分析成了必然的选择。

目前，已经有一些比较成熟的神经网络软件包，其中，应用最广泛的软件包之一就是 MATLAB 的神经网络工具箱。自从 MATLAB5.x 提供了该工具箱以来，它已成为工程人员进行神经网络分析与设计的首选，该工具箱随着 MATLAB 版本的发展，自身也不断完善

与提高，变得越来越全面。

目前神经网络工具箱中的内容非常丰富，包含了很多现有的神经网络新成果，其中涉及如下几种网络模型。

- 感知器模型；
- 线性滤波器；
- BP 网络模型；
- 控制系统网络模型；
- 径向基网络模型；
- 自组织网络；
- 反馈网络；
- 自适应滤波和自适应训练。

另外，该工具箱中还包含大量的演示实例，有助于初学者加深理解。它所介绍的网络模型和实例可以通过 MATLAB 的帮助进行学习，前提是要具备一定的英语水平。如在 MATLAB 的命令窗口中输入 **help nnet**，即可得到神经网络工具箱的有关版本信息及函数列表，如

```
>> help nnet
    Neural Network Toolbox
    Version 6.0.2 (R2009a) 15-Jan-2009

    Graphical user interface functions.
       ……
    Analysis functions.
       ……
    Distance functions.
       ……
    Formatting data.
       ……
    Initialize network functions.
      ……
    Initialize layer functions.
       ……
    Initialize weight and bias functions.
       ……
    Learning functions.
       ……
    Line search functions.
       ……
    Net input functions.
       ……
    Network creation functions.
       ……
    Network transform functions.
       ……
    Network update functions.
       ……
    Performance functions.
       ……
```

```
    Plotting functions.
      ……
    Processing data.
      ……
    Simulink support.
      ……
    Topology functions.
      ……
    Training functions.
      ……
    Transfer functions.
      ……
    Using networks.
      ……
    Weight functions.
      ……
    Template custom functions.
      ……
    Contents of nnet:
      ……
```

其中给出了该神经网络工具箱的版本信息，以及图形用户接口函数、分析函数等各类函数，限于篇幅，这里不将各类函数一一列出。神经网络工具箱中准备了丰富的工具函数。其中一些函数式特别针对某一种类型的神经网络的，如感知器的创建函数、BP 网络的训练函数等；另外一些函数则是通用的，几乎可以用于所有类型的神经网络，如神经网络仿真函数、初始化函数和训练函数等。

这里主要介绍了通用的神经网络工具函数，对它们的功能、调用格式、使用方法及注意事项做了详尽说明。表 10-1 列出了神经网络中一些比较重要的通用函数。

表 10-1　通用函数

函 数 类 别	函 数 名 称	函 数 用 途
神经网络仿真函数	sim	针对给定的输入，得到网络输出
神经网络训练函数	train	调用其他训练函数，对网络进行训练
	trainb	对权值和阈值进行训练
	adapt	自适应函数
神经网络学习函数	learnp	网络权值和阈值的学习
	init	对网络进行初始化
	initlay	多层网络的初始化
	initnw	利用 Nguyen-Widrow 准则对层进行初始化
	initwb	调用指定的函数对层进行初始化
神经网络输入函数	netsum	输入求和函数
	netprod	输入求积函数
	concur	使权值向量和阈值向量的结构一致
传递函数	harlim	硬限幅函数
	hardims	对称硬限幅函数
其他	dotprod	权值求积函数

10.2.1 神经网络仿真函数 sim

该函数用于对神经网络进行仿真，调用格式为

```
[Y,Pf,Af,E,perf] = sim(net,P,Pi,Ai,T)
[Y,Pf,Af,E,perf] = sim(net,{Q TS},Pi,Ai,T)
[Y,Pf,Af,E,perf] = sim(net,Q,Pi,Ai,T)
```

函数中各参数意义如下。

Y：函数返回值，网络输出；

Pf：函数返回值，最终输出延迟；

Af：函数返回值，最终的层延迟；

E：函数返回值，网络误差；

perf：函数返回值，网络性能；

net：待仿真的神经网络；

P：网络输入；

Pi：初始输入延迟，默认为 0；

Ai：初始的层延迟，默认为 0；

T：网络目标，默认为 0；

Pi、Ai，Pf 和 Af 是可选的，它们只用于存在输入延迟和层延迟的网络。函数中用到的信号参数采取了两种不同的形式进行表示，分别为单元阵列和矩阵的形式。其中单元阵列能够方便地对多输入多输出的神经网络进行描述。信号参数为单元阵列下的输入/输出形式如下。

P：Ni×TS 维的单元阵列，每个元素 P{i,ts}都是一个 Ri×Q 的矩阵；

Pi：Ni×ID 维的单元阵列，每个元素 P{i,k}都是一个 Ri×Q 的矩阵；

Ai：Nl×LD 维的单元阵列，每个元素 Ai{i,k}都是一个 Si×Q 的矩阵；

T：Nt×TS 维的单元阵列，每个元素 P{i,ts}都是一个 Vi×Q 的矩阵；

Y：No×TS 维的单元阵列，每个元素 Y{i,ts}都是一个 Ui×Q 的矩阵；

Pf：Ni×ID 维的单元阵列，每个元素 Pf{i,k}都是一个 Ri×Q 的矩阵；

Af：Nl×LD 维的单元阵列，每个元素 Af{i,k}都是一个 Si×Q 的矩阵；

E：Nt×TS 维的单元阵列，每个元素 P{i,ts}都是一个 Vi×Q 的矩阵。

其中，

Ni：神经网络输入的数目；

Nl：神经网络层次的数目；

No：神经网络输出的数目；

ID：输入延迟的数目；

LD：层次延迟的数目；

TS：时间步长的数目；

Q：批量；

Ri：第 i 个输入的长度；

Si：第 i 层的长度；

Ui：第 i 个输出的长度；

Pi{i,k}：第 i 个输入在 ts = k – ID 时刻的状态；

Pf{i,k}：第 i 个输入在 ts = TS + k – ID 时刻的状态；

Ai{i,k}：某层输出 i 在 ts = k – ID 时刻的状态；

Af{i,k}：某层输出 i 在 ts = TS + k – ID 时刻的状态。

矩阵的形式只用于仿真的时间步长 TS = 1 的场合，它适用于单输入单输出的网络，但也同样适用于多输入多输出的网络。在矩阵形式下，每个矩阵都是将对应的单元阵列中的元素组合而成的。其中，

P：Ri 的总和×Q 维矩阵；

Pi：Ri 的总和×(ID×Q) 维矩阵；

Ai：Si 的总和×(LD×Q) 维矩阵；

T：Vi 的总和×Q 维矩阵；

Y：Ui 的总和×Q 维矩阵；

Pf：Ri 的总和×(ID×Q) 维矩阵；

Af：Si 的总和×(LD×Q) 维矩阵；

E：Vi 的总和×Q 维矩阵。

10.2.2 神经网络训练及学习函数

（1）train：该函数用于对神经网络进行训练，调用格式为：

```
[net,tr,Y,E,Pf,Af] = train(NET,P,T,Pi,Ai)
[net,tr,Y,E,Pf,Af] = train(NET,P,T,Pi,Ai,VV,TV)
```

各参数的意义如下所示。

NET：待训练的神经网络；

P：网络的输入信号；

T：网络的目标，默认为 0；

Pi：初始的输入延迟，默认为 0；

Ai：初始的层次延迟，默认为 0；

VV：网络结构确认向量，默认为空；

TV：网络结构测试向量，默认为空；

net：函数返回值，训练后的神经网络；

tr：函数返回值，训练记录（包括步长和性能）；

Y：函数返回值，神经网络输出信号；

E：函数返回值，神经网络误差；

Pf：函数返回值，最终输入延迟；

Af：函数返回值，最终层延迟。



需要指出的是，参数 T 是可选的。只有当需要明确神经网络的目标时，才调用该参数。同样，Pi 和 Pf 也是可选的，它们只用于存在输入延迟和层延迟的场合。VV 和 TV 也是可选的，而且它们除了采用空矩阵之外，只能从以下范围中取值。

VV.P/TV.P：确认/测试输入信号；

VV.P/TV.T：确认/测试目标，默认为 0；

VV.Pi/TV.Pi：确认/测试初始的输入延迟，默认为 0；

VV.Ai/VV.Ai：确认/测试层确认延迟，默认为 0。



调用该函数对网络进行训练之前，需要首先设定实际的训练函数，如 trainlm 或 traindx 等，然后该函数调用相应的算法对网络进行训练，也就是说，函数 train 只是调用设定的或默认的训练函数对网络进行训练。

（2）trainb：该函数用于神经网络权值和阈值的训练，调用格式为：

```
[NET,TR,Ac,El] = train(net,Pd,Tl,Ai,Q,TS,VV,VT)
info = trainb(code)
```

各参数的意义如下所示。

net：待训练的神经网络；

Pd：已延迟的输入信号；

Tl：层目标；

Ai：初始的输入；

Q：批量；

TS：时间步长；

VV：确认向量或者为空矩阵；

TV：测试向量或者为空矩阵；

NET：函数返回值，训练后的神经网络；

TR：函数返回值，每一步的训练记录如下：

① TR.epoch——仿真步次；

② TR.perf——训练性能；

③ TR.vperf——确认性能；

④ TR.tperf——测试性能。

Ac：训练停止后，聚合层的输出；

El：训练停止后的层误差。

另外，训练之前需要设定以下参数，默认值如表 10-2 所示。

表 10-2　训练参数

训练参数名称	默 认 值	属 性
net.trainParam.epochs	100	最大训练步数
net.trainParam.goal	0	性能参数
net.trainParam.max_fail	5	确认失败的最大次数
net.trainParam.show	25	两次显示之间的训练步数（无显示时取 NaN）
net.trainParam.time	inf	最大训练时间（单位：s）



该函数并不能被直接调用，而是通过函数 train 隐含调用，train 通过设置网络属性 NET.trainFcn 为“trainb”来调用 trainb 对网络进行训练。

（3）learnp：该函数用于神经网络权值和阈值的学习，调用格式为：

```
[dW,LS] = learnp(W,P,Z,N,A,T,E,gA,D,LP,LS)
[db,LS] = learnp(b,ones(1,Q),Z,N,A,T,E,gW,gA,D,LP,LS)
info = learnp(code)
```

各参数的意义如下所示。

W：$S \times R$ 维的权值矩阵（或 $S \times 1$ 维的阈值向量）；

P：Q 组 R 维的输入向量（或 Q 组单个输入）；

Z：Q 组 S 维的权值输入向量；

N：Q 组 S 维的网络输入向量；

A：Q 组 S 维的输出向量；

T：Q 组 S 维的目标向量；

E：Q 组 S 维的误差向量；

gW：$S \times R$ 维的性能参数的梯度；

gA：Q 组 S 维的性能参数的输出梯度；

LP：学习参数，若没有则为空；

LS：学习状态，初始值为空；

dW：$S \times R$ 维权值（或阈值）的变化矩阵；

LS：新的学习状态。

info = learnp(code)中 code 的可取值为：

① pname——返回学习的名称；

② pdefaults——返回默认的学习参数；

③ needg——如果函数使用了 gW 或 gA，则返回 1。

（4）learnpn：该函数也是一个权值和阈值的学习函数，但它输入向量的幅值变化非常大，存在奇异值时，其学习速度比 learnp 要快得多，其调用格式为：

```
[dW,LS] = learnpn(W,P,Z,N,T,E,gW,gA,D,LP,LS)
info = learnpn(code)
```

其中，参数含义与 learnp 函数相同。

（5）adapt：该函数使得神经网络能够自适应，调用格式为：

```
[net,Y,E,Pf,Af,tr] = adapt(NET,P,T,Pi,Ai)
```

参数意义如下所示。

NET：未适应的神经网络；

P：网络输入；

T：网络目标，默认为 0；

Pi：初始输入延迟，默认为 0；

Ai：初始层延迟，默认为 0。

通过设定自适应的参数 net.adaptParam 和自适应的函数 net.adaptFunc 可调用该函数，

并返回如下参数。

net：自适应后的神经网络；

Y：网络输出；

E：网络误差；

Pf：最终输入延迟；

Af：最终层延迟；

tr：训练记录（步数和性能）。



参数 T 是可选的，并且只用于必须指明网络目标的场合，同样，Pi 和 Pf 也是可选的，它们只用于存在输入延迟和层延迟的网络。

（6）revert：该函数用于将更新后的权值和阈值恢复到最后一次初始化的值，调用格式为：

```
net = revert(net)
```

如果网络结构已经发生了变化，也就是说，如果网络的权值和阈值之间的连接关系及输入、每层的长度都与原来的网络结构有所不同，那么，该函数无法将权值和阈值恢复到原来的值，在这种情况下，函数将权值和阈值都设置为 0。

10.2.3 神经网络初始化函数

（1）init：该函数用于对神经网络进行初始化，调用格式为：

```
NET = init(net)
```

参数意义如下所示。

NET：返回参数，表示已经初始化后的神经网络；

net：待初始化的神经网络。

NET 为 net 经过一定的初始化修正而成，修正后，前者的权值和阈值都发生了改变。

（2）initlay：该函数特别使用于层——层结构的神经网络的初始化，调用格式为：

```
NET = initlay(net)
Info = initlay(code)
```

各参数意义如下所示。

net：待初始化的神经网络。

NET：初始化后的神经网络。

Info = initlay(code)：根据不同的 code 代码返回不同的信息，code 可为以下两项。

① pname——返回初始化参数的名称。

② pdefaults——返回默认的初始化参数。

通过指定神经网络每一层 i 的初始化函数 NET.layers[i]来调用该函数，初始化后的神经网络每一层都得到了修正。

（3）initnw：该函数是一个层初始化函数，它按照 Nguyen-Widrow 准则对某层的权值和阈值进行初始化，调用格式为：



```
NET = initnw(net,i)
```

参数意义如下所示。

net：待初始化的神经网络；

i：层次索引；

NET：初始化后的神经网络。

（4）initwb：该函数也是一个层初始化函数，它按照设定的每层的初始化函数对每层的权值和阈值进行初始化，调用格式为：

```
NET = initwb(net,i)
```

其中，参数的意义如下所示。

net：待初始化的神经网络；

i：层次索引；

NET：初始化后的神经网络。

【例 10-1】 根据给定的输入向量 P 和目标向量 T，创建一个感知器网络，对其进行训练并初始化，其中，P＝[0 1 0 1;0 0 1 1]，T＝[0 0 0 1]。

代码如下。

```
%Matlab Program eg10 1.m
net=newp([0 1;-2 2],1);
%感知器的 P 和阈值
net.iw{1,1}
net.b{1}
P = [0 1 0 1;0 0 1 1];
T = [0 0 0 1];
%对感知器进行训练
net = train(net,P,T);
net.iw{1,1}
net.b{1}
net = init(net);
net.iw{1,1}
net.b{1}
```

训练过程如图 10-2 所示。

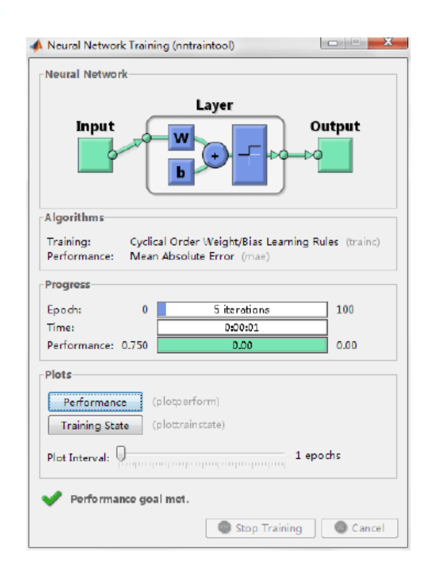


图 10-2　训练过程



运行结果如下。

```
ans =
        0    0
ans =
        0
ans =
        1    2
ans =
        -3
ans =
        0    0
ans =
        0
```

由此可见，在感知器刚刚创建，尚未训练以前，权值和阈值都为 0；训练以后，权值和阈值分别为[1 2]和-3；对训练好的网络进行初始化后，恢复到以前的值，都为 0。

10.2.4　神经网络输入函数

（1）netsum：该函数是一个输入求和函数，它通过将某一层的加权输入和阈值相加作为该层的输入，调用格式为：

```
N = netsum(Z1,Z2,…)
df = netsum('deriv')
```

其中，参数的意义如下所示。

Zi(i=1,2,3,…)：第 i 个输入，它的数目可以是任意个；

df=netsum('deriv')：返回的是 netsum 的微分函数 dnetsum。

（2）netprod：与 netsum 的计算框架类似，不过该函数是输入求积函数，它将某一层的权值和阈值相乘作为该层的输入，调用格式为

```
N=netprod(Z1,Z2,…)
df=netprod('deriv')
```

其中，参数的意义如下所示。

Zi(i=1,2,3,…)：第 i 个输入，它的数目可以是任意个；

df=netsum('deriv')：返回的是 netsum 的微分函数 dnetsum。

（3）concur：该函数的作用在于使得本来不一致的权值向量和阈值向量的结构一致，以便于进行相加或相乘运算，调用格式为：

```
concur(b,q)
```

其中，各参数意义如下所示。

b：N1×1 维的权值向量；

q：要达到一致化所需要的长度。

返回值为一个已经一致化了的矩阵。

【例 10-2】 有两个加权输入向量，Z1 和 Z2，试调用函数 netsum 将两者相加，调用 netprod 将两者相乘。

MATLAB 代码如下。

```
%Matlab Program eg10_2.m
z1 = [1 2 4;3 4 1];
z2 = [-1 2 2;-5 -6 1];
b = [0;-1];
con1 = concur(b,3)
n1 = netsum(z1,z2)
n2 = netprod(z1,z2)
n3 = netsum(z1,z2,con1)
```

运行结果如下。

```
con1 =
        0     0     0
       -1    -1    -1
n1 =
        0     4     6
       -2    -2     2
n2 =
       -1     4     8
       -15   -24     1
n3 =
        0     4     6
       -3    -3     1
```

由此可以看出，netsum 和 netprod 的运算规则就是将权值和阈值向量中对应的元素相加或相乘，同时也显示了函数 concur 的运行机理，即修正后的矩阵就是由向量的副本组合而成的。

10.2.5 神经网络传递函数

传递函数的作用是将神经网络的输入转换为输出。

（1）hardlim：该函数为硬限幅传递函数，调用格式为：

```
A = hardlim(N)
Info = hardlim(code)
```

参数意义如下所示。

N：某层的 Q 组 S 维的输入向量。

A：返回该层的输出向量。当 N>0 时，返回值为 1；当 N<0 时，返回值为 0。

Info = hardlim(code)：根据不同的 code 代码返回不同的信息，包括如下信息。

① deriv——导数函数名称；

② name——传递函数的全称；

③ output——传递函数的输出范围；

④ active——传递函数的输入范围。

该函数的原型函数为

$$\text{hard}\lim(n)\begin{cases}1 & n\geqslant 0\\0 & n<0\end{cases}$$



可通过将 NET.layers{i}.transferFcn 设定为 'hardlim' 来调用该函数，这设置了神经网络中第 i 层的传递函数。

（2）hardlims：该函数为对称的硬限幅传递函数，调用格式为：

```
A = hardlims(N)
Info = hardlims(code)
```

其中，参数的意义如下。

N：某层的 Q 组 S 维的输入向量。

A：函数返回值。当 N>0 时，返回值为 1；当 N<0 时，返回值为–1。

Info = hardlims(codes)的含义参见 hardlim。

该函数的原型函数为

$$\text{hard lims}(n)\begin{cases}1 & n\geqslant 0\\-1 & n<0\end{cases}$$

由此可见，以上两个函数可以实现神经网络的分类和判断功能。

【例 10-3】利用 MATLAB 给出一个数组，调用函数 hardlim 与 hardlims 对其进行分类。Matlab 代码如下。

```
%Matlab Program eg10 3.m
n = -5:0.1:5
a = hardlim(n);
b = hardlims(n);
plot(n,a,'bo');
hold on;
plot(n,b,'r');
hold on;
```

代码中的函数，可参照 MATLAB 帮助进行学习，这里不再赘述，命令窗口显示结果如下。

```
n =
   Columns 1 through 10
   -5.0000  -4.9000  -4.8000  -4.7000  -4.6000  -4.5000  -4.4000  -4.3000
   -4.2000  -4.1000
   Columns 11 through 20
   -4.0000  -3.9000  -3.8000  -3.7000  -3.6000  -3.5000  -3.4000  -3.3000
   -3.2000  -3.1000
   Columns 21 through 30
   -3.0000  -2.9000  -2.8000  -2.7000  -2.6000  -2.5000  -2.4000  -2.3000
   -2.2000  -2.1000
   Columns 31 through 40
   -2.0000  -1.9000  -1.8000  -1.7000  -1.6000  -1.5000  -1.4000  -1.3000
   -1.2000  -1.1000
```

```
Columns 41 through 50
-1.0000  -0.9000  -0.8000  -0.7000  -0.6000  -0.5000  -0.4000  -0.3000
-0.2000  -0.1000
Columns 51 through 60
      0    0.1000   0.2000   0.3000   0.4000   0.5000   0.6000   0.7000
 0.8000   0.9000
Columns 61 through 70
 1.0000   1.1000   1.2000   1.3000   1.4000   1.5000   1.6000   1.7000
 1.8000   1.9000
Columns 71 through 80
 2.0000   2.1000   2.2000   2.3000   2.4000   2.5000   2.6000   2.7000
 2.8000   2.9000
Columns 81 through 90
 3.0000   3.1000   3.2000   3.3000   3.4000   3.5000   3.6000   3.7000
 3.8000   3.9000
Columns 91 through 100
 4.0000   4.1000   4.2000   4.3000   4.4000   4.5000   4.6000   4.7000
 4.8000   4.9000
Column 101
 5.0000
```

其运行结果如图 10-3 所示。图中实点部分是函数 hardlims 的分类结果；圆圈部分是 hardlim 的分类结果。可以看出，两个函数都成功地对数组进行了分类。

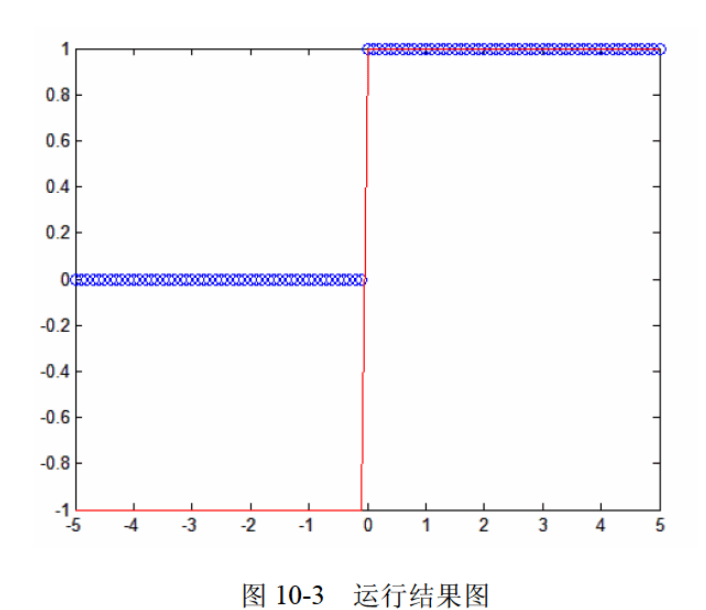


图 10-3　运行结果图

10.2.6　其他重要函数

dotprod：该函数用于对权值求点积，它求得权值与输入之间的点积作为加权输入，调用格式为：

```
Z = dotprod(W,P)
df = dotprod('deriv')
```

其中，参数的意义如下所示。

W：$S \times R$ 维的权值矩阵；

P：Q 组 R 维的输入向量；

Z：Q 组 R 维的 W 与 P 的点积；

df = dotprod('deriv')：返回函数的导数。

【例 10-4】 建立两个矩阵（或向量），演示函数 dotprod 的功能。

MATLAB 代码如下。

```
%Matlab Program eg10 4.m
W = rand(4,3)
P = rand(3,1)
Z = dotprod(W,P)
```

运行结果为

```
W =
    0.8147    0.6324    0.9575
    0.9058    0.0975    0.9649
    0.1270    0.2785    0.1576
    0.9134    0.5469    0.9706
P =
    0.9572
    0.4854
    0.8003
Z =
    1.8530
    1.6865
    0.3829
    1.9164
```

对于神经网络的其他函数，本章不再一一说明，读者可根据需要，通过 help 命令查询获得帮助信息。

10.3 模糊逻辑工具箱

针对模糊逻辑尤其是模糊控制的迅速推广应用，MathWorks 公司在其 MATLAB 版本中添加了模糊逻辑工具箱（Fuzzy Logic Toolbox），MATLAB 模糊逻辑工具箱提供了建立和测试模糊逻辑系统的一整套功能函数，包括定义语言变量及其隶属度函数、输入模糊推理规则、整个模糊推理系统（FIS）的管理及交互式地观察模糊推理的过程和输出结果。本节将对其图形用户界面及命令行工作方式进行讲解。

10.3.1 MATLAB 模糊逻辑工具箱的图形用户界面

MATLAB 模糊逻辑工具箱提供了 GUI（图形用户接口）工具，方便用户直观地编辑模

糊推理系统，图形用户界面主要包括以下几个工具。

- ❑ **模糊推理系统编辑器（FIS Editor）** 用于建立模糊逻辑系统的整体框架，包括输入输出数目、去模糊化方法等。
- ❑ **隶属度函数编辑器（Membership Function Editor）** 用于建立语言变量的隶属度函数。
- ❑ **模糊推理规则编辑器（Rule Editor）** 用于建立模糊规则。
- ❑ **系统输入输出特性曲面浏览器（Surface Viewer）** 用于观察模糊推理图。

下面将详细介绍这几个主要工具的使用方法。

1. FIS 编辑器（FIS Editor）

在 MATLAB 中可以通过以下两种方式自启动模糊推理系统编辑器（FIS Editor）。

（1）在命令窗口中输入 fuzzy：

```
>>fuzzy
```

（2）在 MATLAB 主窗口左下角选择【Start】/【Toolboxes】/【Fuzzy Logic】/【FIS Editor Viewer】命令。

用上述方法打开模糊推理系统编辑器，其界面如图 10-4 所示。

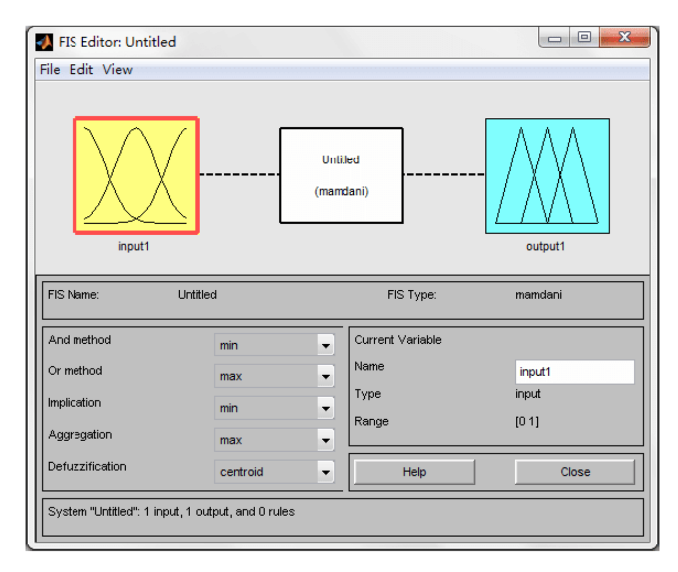


图 10-4 FIS 编辑器

在 FIS 编辑器窗口的上半部分有 3 个组成部分：input1、Untitled（mamdani）和 output1。双击 input1 上方的图形，将打开隶属度函数编辑器（Membership Function Editor）窗口，快捷键为 Ctrl+2；双击 Untitled（mamdani）上方的图形，将打开模糊推理规则编辑器（Rule Editor）窗口，快捷键为 Ctrl+3；双击 output1 上方的图形，将同样打开隶属度函数编辑器窗口。

在 FIS 编辑器窗口的下半部分，用户可以观察到模糊推理系统的名称（FIS Name）和类型（FIS Type）。在下半部分左侧，窗口中列出了模糊推理系统的一些基本属性，包括 And method（与运算）、Or method（或运算）、Implication（蕴含运算）、Aggregation（模糊规则综合运算）和 Defuzzification（去模糊化）。用户可以在这些属性的下拉菜单中选择适合自己的方法，如极小运算 min、极大运算 max 等。在下半部分右侧，用户可以对输入变量或输出变量进行命名或改名。具体操作方法是先单击 input1 上方图形，然后在文本框

Name 后输入期望的变量名即可，输出变量的操作方法类似于输入变量，不再重复。

FIS Editor 窗口系统菜单主要功能叙述如下。

1）File 菜单

【New FIS】/【Mamdani】：新建一个 Mamdani 型模糊推理系统，快捷键为 Ctrl+N。

【New FIS】/【Sugeno】：新建一个 Sugeno 型模型模糊推理系统。

【Import】/【From Workspace】：从 MATLAB 的 Workspace 加载一个模糊推理系统。

【Import】/【rom Disk】：从硬盘加载一个模糊推理系统文件。

【Export】/【To Workspace】：将当前模糊推理系统输出到 Matlab 的 Workspace。

【Export】/【To Disk】：将当前模糊推理系统保存到硬盘。

Print：打印当前模糊推理系统的信息。

Close：关闭当前窗口。

2）Edit 菜单

Undo：取消上次操作，快捷键是 Ctrl+Z。

【Add Variable】/【Input】：添加输入语言变量。

【Add Variable】/【Output】：添加输出语言变量。

Remove Selected Variable：删除所选择的语言。

Membership Function：打开隶属度函数编辑器。

Rules：打开模糊规则编辑器。

3）View 菜单

Rules：打开模糊罪规则浏览器。

Surface：打开模糊系统输入输出特性曲面浏览器。

2．隶属度函数编辑器（Membership Function Editor）

在 MATLAB 中可以通过以下 3 种方式启动隶属度编辑器。

（1）在命令窗口中键入 mfedit

```
>>mfedit
```

（2）在 FIS 编辑器窗口中双击输入图形或输出图形。

（3）在 FIS 编辑器窗口中使用快捷键 Ctrl+F2。

打开的窗口如图 10-5 所示。

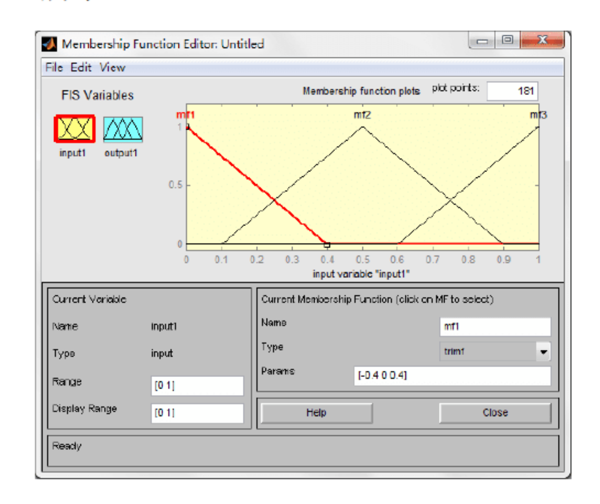


图 10-5　隶属度函数编辑器

在隶属度函数编辑器窗口上方是图形显示区，单击左上方输入变量 input1 或输出变量 output1 图标，可以在输入变量和输出变量的隶属度函数之间切换。右侧则是绘图区，用户可以选中一条曲线，拖动曲线上的圆圈来调整曲线形状和参数，在文本框 plot points 中可以设置绘制图形所用的点数。

在窗口下方则是参数设置区。

在 Range 文本框中，可以设置当前变量的取值范围。

在 Display Range 文本框中，用户可以选择在图形窗口中所显示变量的范围。

在 Name 文本框中，用户可以输入期望的隶属度函数的名称。

在 Type 下拉列表中，用户可以改变当前隶属度函数的类型，包括 trimf、trapmf、gbellmf、gaussmf、gauss2mf、sigmf、dsigmf、psigmf、pimf、smf 和 zmf。

在 Params 文本框中，用户可以设置当前隶属度函数的参数值。

3. 模糊推理规则编辑器（Rule Editor）

在 MATLAB 中可以通过以下几种方式启动模糊推理规则编辑器。

（1）在命令窗口中键入 ruleedit：

```
>>ruleedit
```

（2）在 FIS 编辑器窗口中双击模糊规则图标（如 Untitled (mamdani)）。

（3）在 FIS 编辑器窗口中选择【EDIT】/【Rules】命令，打开模糊推理规则编辑器。

（4）在 FIS 编辑器窗口中使用快捷键 Ctrl+3.

打开的窗口如图 10-6 所示。

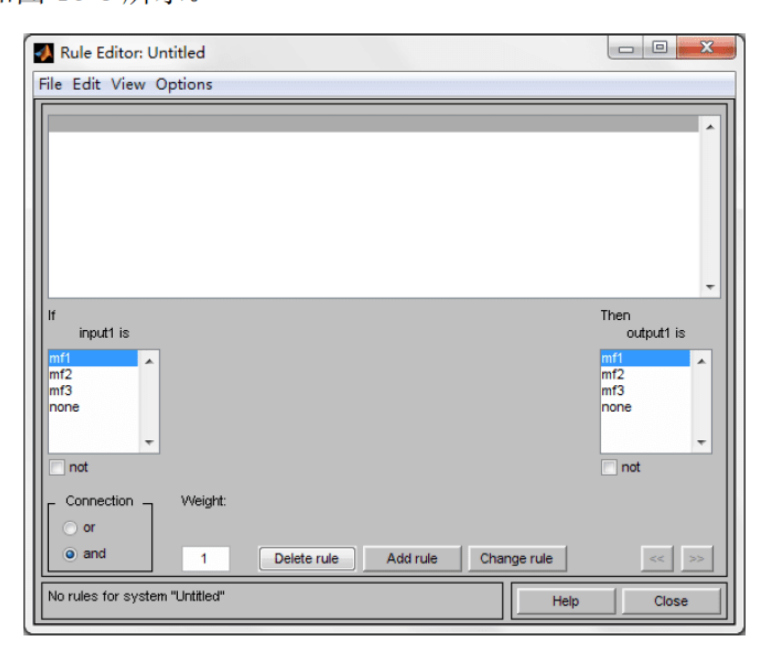


图 10-6　模糊推理规则编辑器

窗口上方是文本编辑窗口，显示用户定义的模糊规则，方便模糊规则的浏览和编辑。模糊规则的通常形式是“IF…THEN…”。

窗口中部是输入输出变量选择窗口 input1 is 和 output1 is，用户可以选择符合需求的输入输出变量来匹配“IF input THEN output”的规则。not 复选框可将模糊规则中的输入输出

表述为“非”。

窗口左下方的 Connection 单选按钮用于选择各输入变量之间的关系是 or（或）还是 and（与）。

窗口下方的 3 个按钮：Delete rule、Add rule 和 Change rule 分别用于删除选中的模糊规则、增加新建的模糊规则和改变选中的模糊规则。

4. 系统输入输出特性曲面浏览器（Surface Viewer）

在 MATLAB 中可以通过以下几种方式启动系统输入输出特性曲面浏览器。

（1）在命令窗口中键入 surfview：

```
>>surfview
```

（2）在 FIS 编辑器窗口中选择【View】/【Surface】命令，打开系统输入特性曲面浏览器。

（3）在 FIS 编辑器窗口中使用快捷键 Ctrl+6。

打开的窗口如图 10-7 所示。

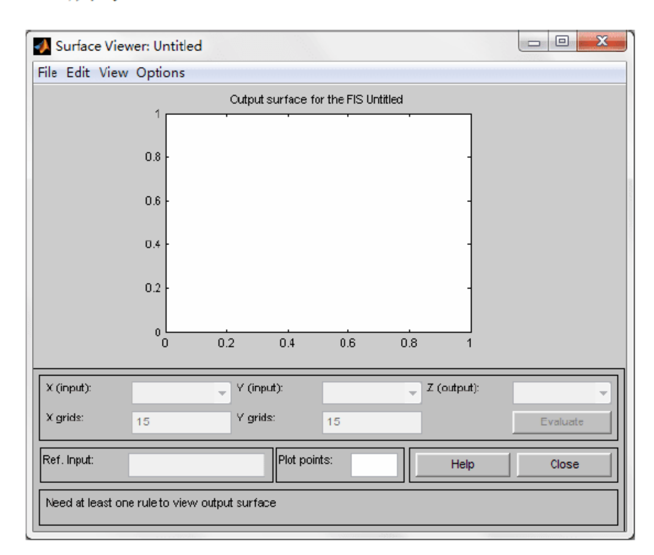


图 10-7　系统输入输出特性曲面浏览器

窗口的上方是图形显示区域，显示系统任意一个输出与任意一个输入或两个输入之间的对应关系。在下拉列表框 X（input）、Y（input）和 Z（output）中，用户可以选择所需显示的变量名称。文本框 X grids 和 Y grids 用于制定输入空间绘图的网格数。文本框 Ref.Input 用于设置输入变量数目大于 2 时其他未显示的输入变量的值。当用户设置了新的输入变量后，可单击窗口右下方的 Evaluate 按钮来计算并绘制新的输出曲面图形。

5. 模糊规则浏览器（Rule Viewer）

在 MATLAB 中可以通过以下几种方式启动模糊规则浏览器（Rule Viewer）。

（1）在命令窗口中键入 ruleview：

```
>>ruleview
```

（2）在 FIS 编辑器窗口中选择【View】/【Rules】命令，打开模糊规则浏览器。

（3）在 FIS 编辑器窗口中使用快捷键 Ctrl+5。

打开的窗口如图 10-8 所示。

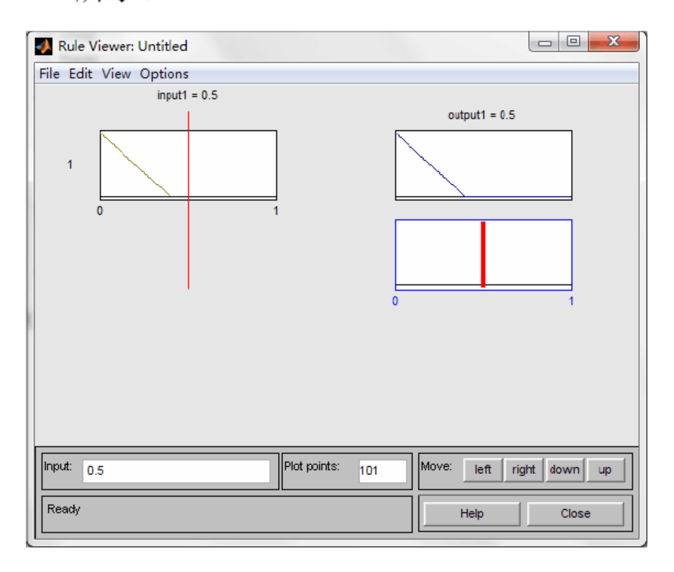


图 10-8 模糊规则浏览器

窗口上方是图形显示区，分别显示输入变量和输出变量在模糊规则中的应用结果，拖动左侧 input1 图形窗口中的数值长线，可以改变输入值，系统自动计算并产生一个新的输出响应，在窗口右侧显示相应的变化。

input 文本框用于设定具体的输入值。

Plot points 文本框用于设定绘制图形的点数。

6. Simulink 工具箱中的模糊逻辑模块

在 MATLAB 中可以通过以下方法打开 Simulink 工具箱中的模糊逻辑模块。

（1）首先打开 Simulink，在浏览器窗口中选择 Fuzzy Logic Toolbox 模块库，如图 10-9 所示。

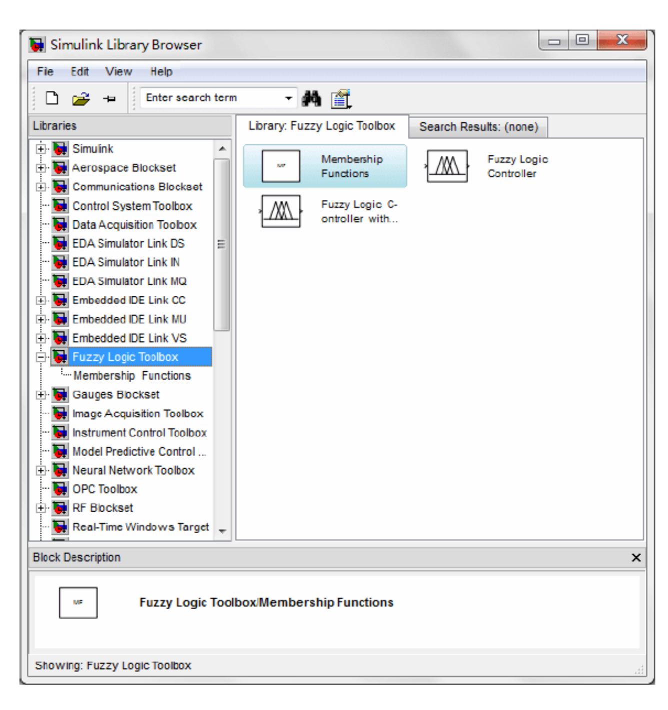


图 10-9 Simulink 工具箱中的模糊逻辑模块

在 Fuzzy Logic Toolbox 模块库中包含有 Fuzzy Logic Controller（模糊逻辑控制器）、Fuzzy Logic Controller with Ruleviewer（带有规则浏览器的模糊逻辑控制器）和 Membership Function（隶属度函数模块库）。

（2）在 MATLAB 命令窗口中输入 fuzblock：

```
>>fuzblock
```

打开 fuzblock 窗口，如图 10-10 所示。

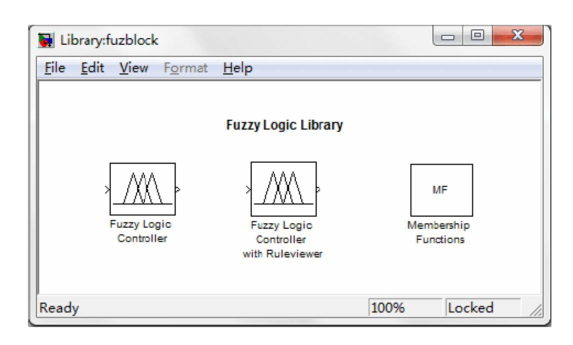


图 10-10　fuzblock 窗口

在如图 10-10 所示的窗口中双击 Membership Function 图标，打开隶属度函数模块库窗口，如图 10-11 所示，该模块库中包含有 Triangular MF、Trapezoidal MF、Generalized Bell MF、Gaussian MF、Sigmoidal MF 等隶属度函数模块。

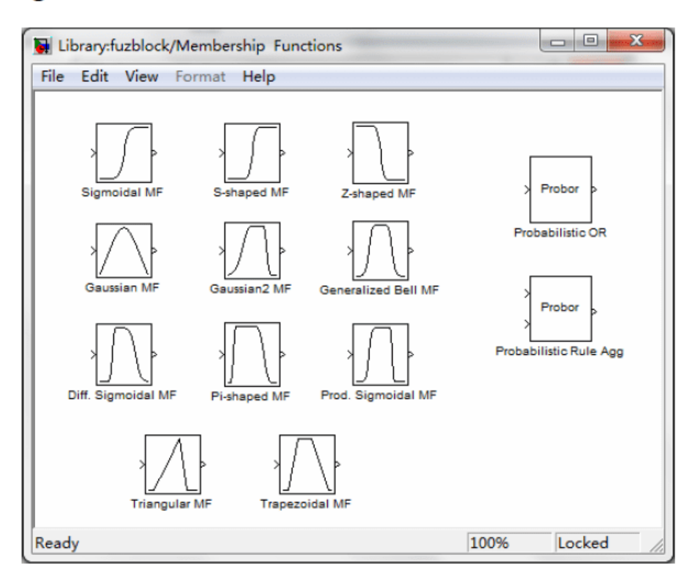


图 10-11　Membership Function 窗口

此外，MATLAB 模糊逻辑工具箱还提供了丰富的演示实例模型，启动这些模型的命令如表 10-3 所示。

表 10-3　Matlab 模糊逻辑工具箱中的演示实例

演示实例命令	演示实例内容
defuzzdm	反模糊化方法
fcmdemo	FCM 聚类显示（二维）

演示实例命令	演示实例内容
fuzdemos	列出所有模糊逻辑工具箱的演示程序
gasdemo	使用子聚类节省燃料的 ANFIS 演示
juggler	带规则观察器的弹球游戏器
invkine	双关节平面机械臂倒立摆运动
irisfcm	FCM 聚类显示（四维）
noisedm	自适应消除噪声
slbb	球和棒控制（Simulink 方式）
slcp	倒立摆控制（Simulink 方式）
sltank	水位控制（Simulink 方式）
sltankrule	带规则观察器的水位控制（Simulink 方式）
sltbu	卡车支援（Simulink 方式）

例如在命令窗口中输入 invkine 命令：

```
>>invkine
```

打开窗口如图 10-12 所示，用户可以在窗口中单击 Start Animation 按钮来启动动画演示。

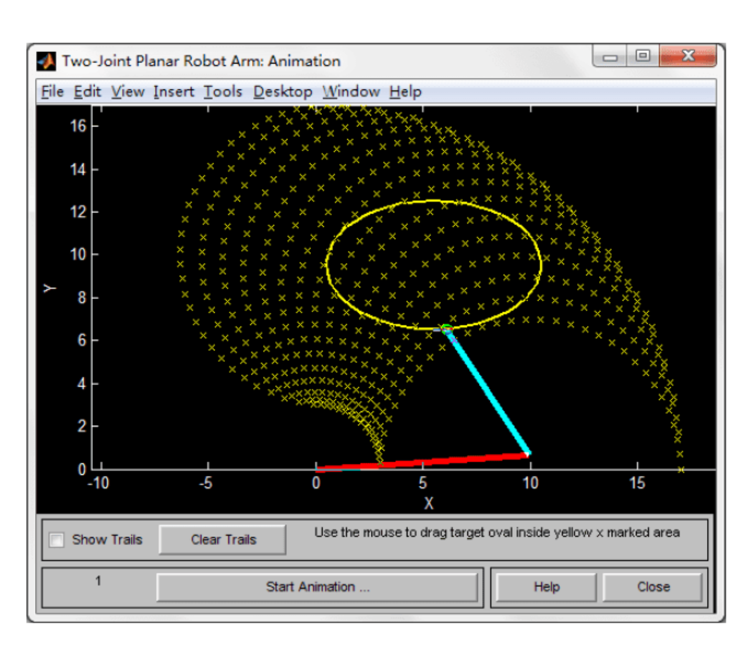


图 10-12　Two-Joint Planar Robot Arm 动画演示窗口

10.3.2　MATLAB 模糊逻辑工具箱的命令行工作方式

1．常用 FIS 工具函数介绍

除了 GUI 方式外，还可以通过命令行的工作方式来使用模糊逻辑工具箱，常用的 FIS 工具函数如表 10-4 所示。

表 10-4　常用 FIS 工具函数

函　数　名	函 数 功 能
addmf()	向 FIS 的语言变量添加隶属度函数
addrule()	向 FIS 的语言变量添加规则
addvar()	向 FIS 添加语言变量
defuzz()	对隶属度函数进行去模糊化
evalfis()	完成模糊推理计算
evalmf()	通过隶属度函数计算
gensurf()	生成一个 FIS 输出曲面
getfis()	得到模糊系统的属性
mf2mf()	在两个隶属度函数之间转换参数
newfis()	产生新的 FIS
parsrule()	解析模糊规则
plotfits()	绘制一个 FIS
plotmf()	绘制给定语言变量的所有隶属度函数的曲线
readfis()	从磁盘装入一个 FIS
rmmf()	从 FIS 中删除某一语言变量的某一隶属度函数
rmvar()	从 FIS 中删除某一语言变量
setfis()	设置模糊系统的属性
showfis()	以分行的形式显示 FIS 结构的所有属性
showrule()	显示 FIS 的规则
writefis()	保存 FIS 到磁盘上

几个主要功能函数的使用方法如下。

（1）newfits()。用于产生新的 FIS，调用格式为：

```
a = newfits(FISNAME,FISTYPE,andMethod,orMethod,impMethod,aggMethod,
defuzzMethod)
```

其中，FISNAME 为 FIS 名称；FISTYPE 为 FIS 类型，如 mamdani；andMethod 为与运算操作符；orMethod 为或运算操作符；impMethod 为蕴含运算操作符；aggMethod 为模糊集合运算操作符；defuzzMethod 为去模糊化方法。返回 a 在 MATLAB 内存中以矩阵形式存储。

（2）addvar()。用于向 FIS 添加语言变量，调用格式为：

```
a = addvar(a,varType,varName,varBounds)
```

其中，a 为 FIS 名称；varType 为变量类型，如 input 和 output；varName 为变量名称；varBounds 为变量范围。

例如：

```
>>a = newfis('tipper');
>>a = addvar(a,'input','service',[0 10]);
```

（3）addmf()。用于向 FIS 的语言变量添加隶属度函数，调用格式为：

```
a = addmf(a,varType,varIndex,mfName,mfType,mfParams)
```

其中，a 为 FIS 名称；varType 为变量类型，如 input 和 output；varIndex 为输入输出变量编号；mfName 为隶属度函数名称；mfType 为隶属度函数类型；mfParams 为隶属度函数参数。

例如：

```
>>a = newfis('tipper');
>>a = addvar(a,'input','service',[0 10]);
>>a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
>>a = addmf(a,'input',1,'good','gaussmf',[1.5 5]);
>>a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
>>plotmf(a,'input',1)
```

得到隶属度函数曲线如图 10-13 所示。

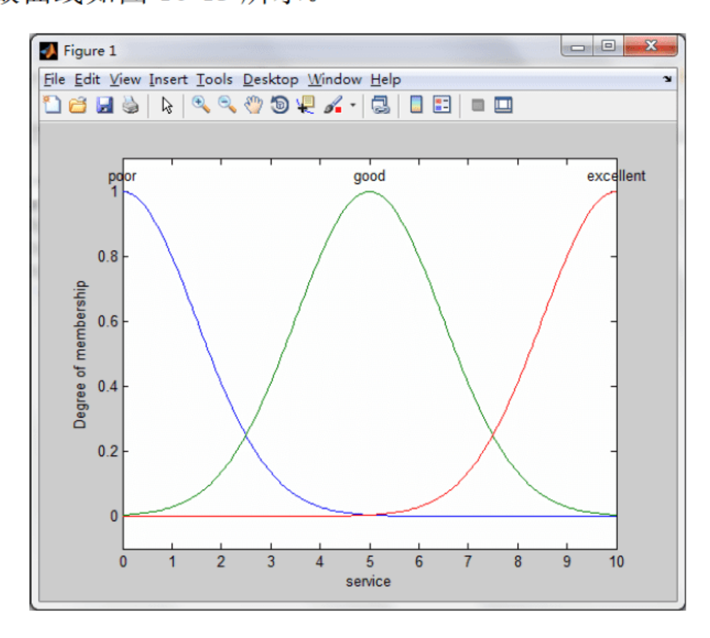


图 10-13　隶属度函数曲线示例

（4）addrule()。用于向 FIS 添加控制规则，调用格式为：

```
a = addrule(a,ruleList)
```

其中，a 为 FIS 名称；ruleList 为变量控制规则表；ruleList 为向量形式。如果模糊推理系统有 m 个输入语言变量和 n 个输出语言变量，则向量 ruleList 的列数必须为 m+n+2，行数任意。在 ruleList 每一行中，前 m 个数字表示各输入语言变量的语言值序号（没有的话用 0 表示），其后 n 个数字表示输出语言变量的语言值序号（没有的话用 0 表示），第 m+n+1 个数字是该规则适用的权重，取值范围在[0,1]之间，通常置为 1。第 m+n+2 个数字表示模糊逻辑运算关系，若控制规则中执行 AND 运算，则置为 1；若控制规则中执行 OR 运算，则置为 2。

如 ruleList = [1 1 2 1 2]表示：“IF input is MF 1 OR input2 is MF, THEN output1 is MF 2”。

（5）evalfis()。用于执行模糊推理计算，调用格式为：

```
Y = evalfis(U,FIS)
```

其中，U 为输入数据，是一个 m×n 矩阵，n 为输入变量数，m 为数据组数；FIS 为模糊推理系统矩阵名称；Y 为输出数据，是一个 m×l 矩阵，l 为输出变量数。

（6）defuzz()。用于对隶属度函数进行去模糊化，调用格式为：

```
out = defuzz(x,mf,type)
```

其中，x 是论域变量范围；mf 为模糊逻辑隶属度函数；type 为去模糊化方法。

MATLAB 中主要有以下 5 种去模糊化方法。

① centroid：面积重心法；

② bisector：面积平分法；

③ mom：最大隶属度平均法；

④ som：最大隶属度取最小值法；

⑤ lom：最大隶属度取最大值法。

例如：

```
>> x = -10:0.1:10;
>> mf = trapmf(x,[-10 -8 -4 7]);
>> xx = defuzz(x,mf,'centroid');
```

（7）writefis()。将 FIS 保存到磁盘上，调用格式为：

```
writefis(FISMAT,'filename')
```

其中，FISMAT 为 FIS 矩阵名称；filename 为设定的文件名。

（8）readfis()。读取磁盘上的 FIS 文件，调用格式为：

```
FISMAT = readfis('filename')
```

其中，filename 为硬盘上的 FIS 文件名；FISMAT 为读取文件数据的 FIS 矩阵名。

（9）getfis()。用于获取模糊推理系统的属性，调用格式为：

```
getfis(FIS)
```

其中，FIS 为 FIS 矩阵名。

例如：

```
>> a = newfis('tipper');
>> a = addvar(a,'input','service',[0 10]);
>> a = addmf(a,'input',1,'poor','gaussmf',[1.5 0]);
>> a = addmf(a,'input',1,'excellent','gaussmf',[1.5 10]);
>> getfis(a)
```

运行结果为：

```
Name = tipper
Type = mamdani
NumInputs = 1
InLabels = service
NumOutputs = 0
```

```
OutLabels =
NumRules = 0
AndMethod = min
OrMethod = max
ImpMethod = min
AggMethod = max
DefuzzMethod = centroid
ans =
    tipper
```

2. 主要隶属度函数类型

Matlab 模糊逻辑工具箱中提供的隶属度函数如表 10-5 所示。

表 10-5 隶属度函数命令

函 数 名	函 数 功 能
gaussmf()	建立高斯型隶属度函数
gauss2mf()	建立双边高斯型隶属度函数
gbellmf()	建立一半的钟型隶属度函数
pimf()	建立 π 型隶属度函数
sigmf()	建立 Sigmiod 型的隶属度函数
trapmf()	建立梯形隶属度函数
trimf()	建立三角型隶属度函数
zmf()	建立 Z 型隶属度函数
mf2mf()	隶属度函数间的参数转换
psigmf()	计算两个 Sigmiod 型隶属度函数之积
dsigmf()	计算两个 Sigmiod 型隶属度函数之和

表 10-5 中的几个主要隶属度函数的使用方法如下。

（1）gaussmf()。建立高斯型隶属度函数，该函数的调用格式为：

```
y = gaussmf(x,params)
y = gaussmf(x,[sigma,c])
```

其中，x 指定变量范围，c 指定函数的中心点，sigma 指定函数曲线的宽度 σ，高斯型函数的形状由 σ 和 c 两个参数决定，高斯函数的表达式如下。

$$y = e^{-\frac{(x-c)^2}{\sigma^2}}$$

例如：

```
>>x = (0:0.1:10)';
>>y1 = gaussmf(x,[0.5 5]);
>>y2 = gaussmf(x,[1 5]);
>>y3 = gaussmf(x,[2 5]);
>>y4 = gaussmf(x,[1 8]);
>>subplot(2,1,1);
>>plot(x,[y1 y2 y3 y4]);
>>y1 = gaussmf(x,[1 2]);
>>y2 = gaussmf(x,[1 4]);
```

```
>>y3 = gaussmf(x,[1 6]);
>>y4 = gaussmf(x,[1 8]);
>>subplot(2,1,2);
>>plot(x,[y1 y2 y3 y4]);
>>set(gcf,'name','gaussmf','numbertitle','off');
```

得到高斯型函数曲线如图 10-14 所示。

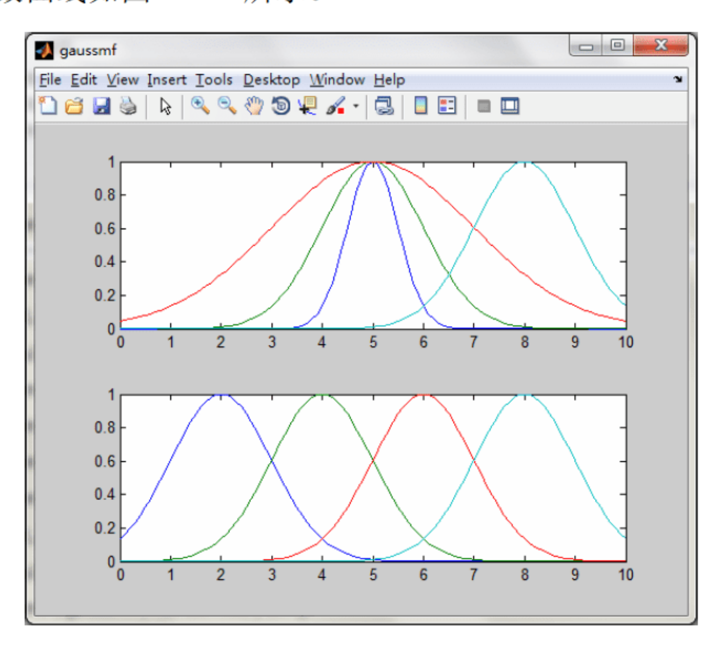


图 10-14　高斯型隶属度函数曲线

（2）gauss2mf()。建立双边高斯型隶属度函数，调用格式为：

```
y = gauss2mf(x,params)
y = gauss2mf(x,[sig1 c1 sig2 c2])
```

双边高斯型函数的曲线由两个中心点不相同的高斯型函数的左右半边曲线组合而成，参数 sig1、c1、sig2、c2 分别对应左右半边高斯函数的宽度与中心点，c2 > c1。数学表达式如下。

$$y=\begin{cases} e^{-\frac{(x-c_1)^2}{\sigma_1^2}}, X<c_1 \\ e^{-\frac{(x-c_2)^2}{\sigma_2^2}} \ X\geqslant c_2 \end{cases}$$

例如：

```
>>x = (0:0.1:10)';
>>y1 = gauss2mf(x,[2 4 1 8]);
>>y2 = gauss2mf(x,[2 5 1 7]);
>>y3 = gauss2mf(x,[2 6 1 6]);
>>y4 = gauss2mf(x,[2 7 1 5]);
>>y5 = gauss2mf(x,[2 8 1 4]);
>>plot(x,[y1 y2 y3 y4 y5]);
>>set(gcf,'name','gauss2mf','numbertitle','off');
```

得到双边高斯型隶属度函数曲线如图 10-15 所示。

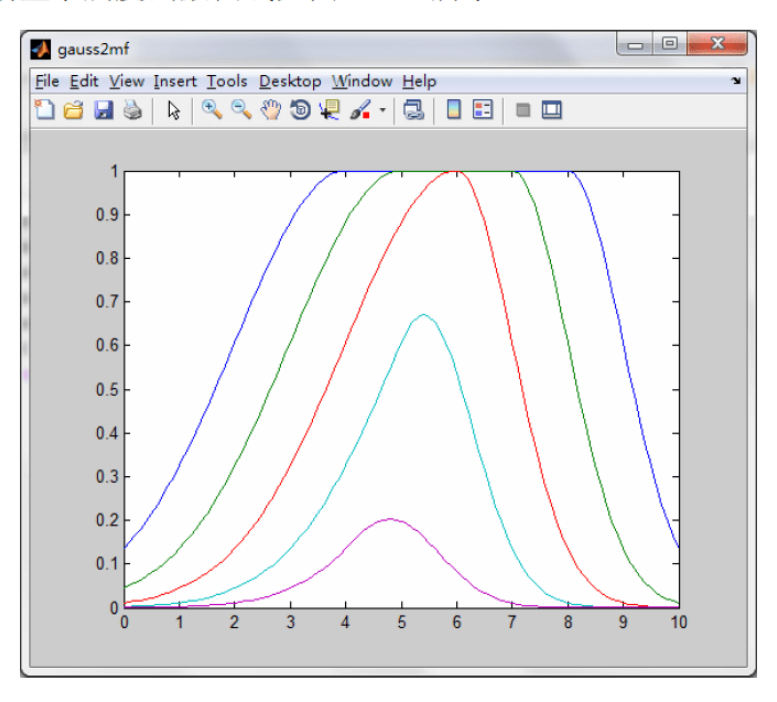


图 10-15　双边高斯型隶属度函数曲线

（3）gbellmf()。建立钟型隶属度函数，调用格式为：

```
y = gbellmf(x,params)
y = gbellmf(x,[a b c])
```

其中，x 指定变量范围；[a b c]指定钟型函数的形状。钟型函数的表达式为：

$$y=\frac{1}{1+\left|\frac{x-c}{a}\right|^{2b}}$$

例如：

```
>>x = (0:0.1:10)';
>>y1 = gbellmf(x,[1 2 5]);
>>y2 = gbellmf(x,[2 4 5]);
>>y3 = gbellmf(x,[3 6 5]);
>>y4 = gbellmf(x,[4 8 5]);
>>subplot(2,1,1);
>>plot(x,[y1 y2 y3 y4]);
>>y1 = gbellmf(x,[2 1 5]);
>>y2 = gbellmf(x,[2 2 5]);
>>y3 = gbellmf(x,[2 4 5]);
>>y4 = gbellmf(x,[2 8 5]);
>>subplot(2,1,2);
>>plot(x,[y1 y2 y3 y4]);
>>set(gcf,'name','gbellmf','numbertitle','off');
```

得到钟型隶属度函数曲线如图 10-16 所示。

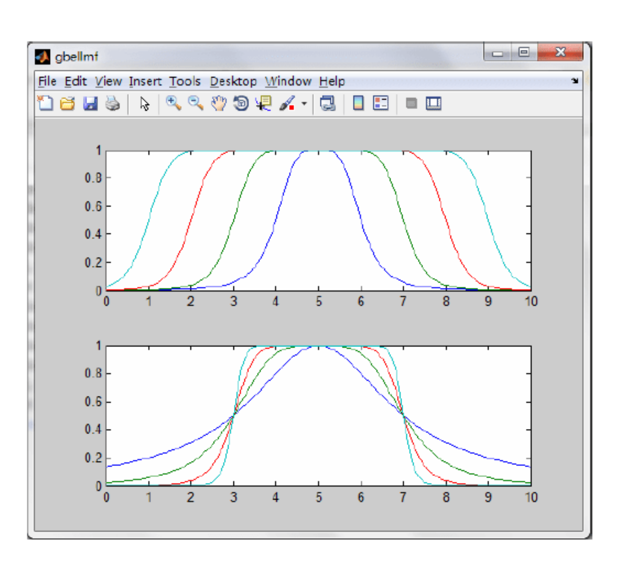


图 10-16　钟型隶属度函数曲线

（4）pimf()。建立 π 型隶属度函数，调用格式为：

```
y = pimf(x,params)
y = pimf(x,[a b c d])
```

其中，x 指定函数的自变量范围；[a b c d]决定函数的形状，a 和 b 分别对应曲线下部的左右两个拐点，d 和 c 分别对应曲线上部的左右两个拐点。

例如：

```
>>x = (0:0.1:10)';
>>y1 = pimf(x,[1 3 9 10]);
>>y2 = pimf(x,[2 5 8 9]);
>>y3 = pimf(x,[3 6 7 8]);
>>y4 = pimf(x,[4 7 6 7]);
>>y5 = pimf(x,[5 8 5 6]);
>>plot(x,[y1 y2 y3 y4 y5]);
>>set(gcf,'name','pimf','numbertitle','off');
```

得到 π 型隶属度函数曲线如图 10-17 所示。

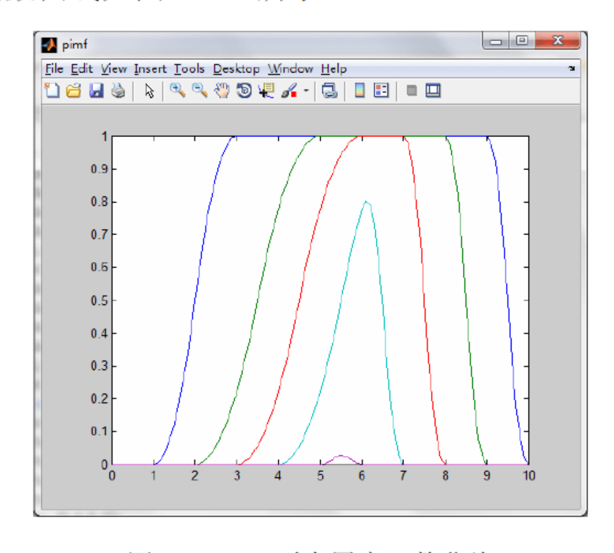


图 10-17　π 型隶属度函数曲线

（5）sigmf()。建立 sigmoid 型隶属度函数，调用格式为：

```
y = sigmf(x,params)
y = sigmf(x,[a c])
```

其中，x 指定函数的自变量范围；[a c]决定了 sigmoid 型函数的形状，其表达式为

$$y=\frac{1}{1+e^{-a(x-c)}}$$

sigmoid 型函数曲线具有半开的形状，因而适于作为“极大”、“极小”等语言值的隶属度函数。

例如：

```
>>x = (0:0.2:10)';
>>y1 = sigmf(x,[-1 5]);
>>y2 = sigmf(x,[-3 5]);
>>y3 = sigmf(x,[4 5]);
>>y4 = sigmf(x,[8 5]);
>>subplot(2,1,1);
>>plot(x,[y1 y2 y3 y4]);
>>y1 = sigmf(x,[5 2]);
>>y2 = sigmf(x,[5 4]);
>>y3 = sigmf(x,[5 6]);
>>y4 = sigmf(x,[5 8]);
>>subplot(2,1,2);
>>plot(x,[y1 y2 y3 y4]);
>>set(gcf,'name','sigmf','numbertitle','off');
```

得到 sigmoid 型函数曲线，如图 10-18 所示。

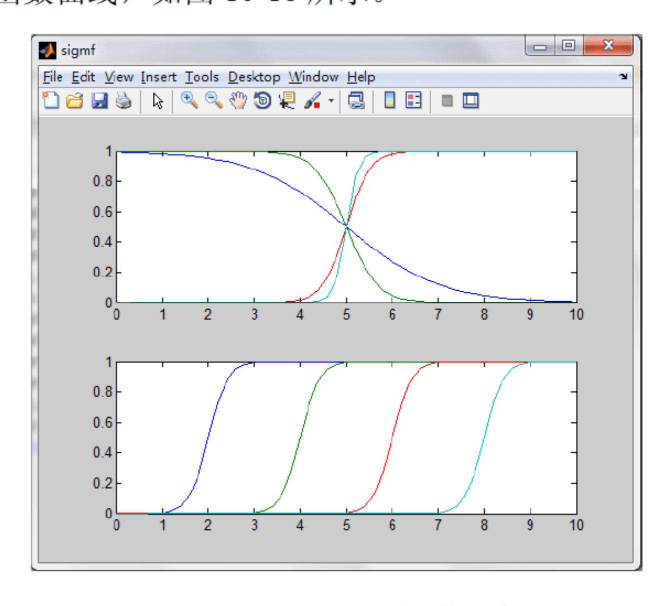


图 10-18　sigmoid 型函数曲线

（6）trapmf()。建立梯形隶属度函数，调用格式为：

```
y = trapmf(x,params)
y = trapmf(x,[a,b,c,d])
```



其中，x 指定函数的自变量范围；参数 a、b、c 和 d 指定梯形隶属度函数的形状，其表达式为

$$y=\begin{cases}0 & x<a\\ \dfrac{x-a}{b-a} & a\leqslant x\leqslant b\\ 1 & b<x<c\\ \dfrac{d-x}{d-c} & c\leqslant x\leqslant d\\ 0 & d<x\end{cases}$$

例如：

```
>>x = (0:0.1:10)';
>>y1 = trapmf(x,[2 3 7 9]);
>>y2 = trapmf(x,[3 4 6 8]);
>>y3 = trapmf(x,[4 5 5 7]);
>>y4 = trapmf(x,[5 6 4 6]);
>>plot(x,[y1 y2 y3 y4]);
>>set(gcf,'name','trapmf','numbertitle','off');
```

得到梯形隶属度函数曲线如图 10-19 所示。

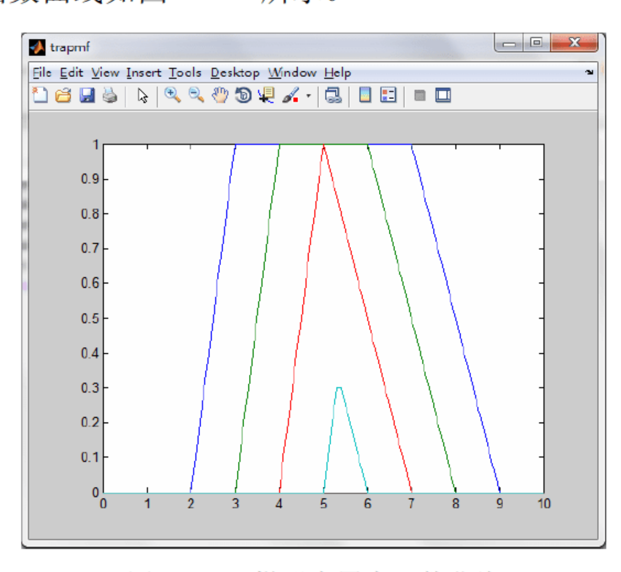


图 10-19　梯形隶属度函数曲线

（7）trimf()。建立三角形隶属度函数，调用格式为：

```
y = trimf(x,params)
y = trimf(x,[a,b,c])
```

其中，x 指定函数的自变量范围；参数 a、b 和 c 指定三角型函数的形状，其表达式为

$$y=\begin{cases}0 & x<a\\ \dfrac{x-a}{b-a} & a\leqslant x\leqslant b\\ \dfrac{c-x}{c-b} & b<x\leqslant c\\ 0 & c<x\end{cases}$$

例如：

```
>>x = (0:0.2:10)';
>>y1 = trimf(x,[3 4 5]);
>>y2 = trimf(x,[2 4 7]);
>>y3 = trimf(x,[1 4 9]);
>>subplot(2,1,1);
>>plot(x,[y1 y2 y3]);
>>y1 = trimf(x,[2 3 5]);
>>y2 = trimf(x,[3 4 7]);
>>y3 = trimf(x,[4 5 9]);
>>subplot(2,1,2);
>>plot(x,[y1 y2 y3]);
>>set(gcf,'name','trimf','numbertitle','off');
```

得到三角形隶属度函数曲线，如图 10-20 所示。

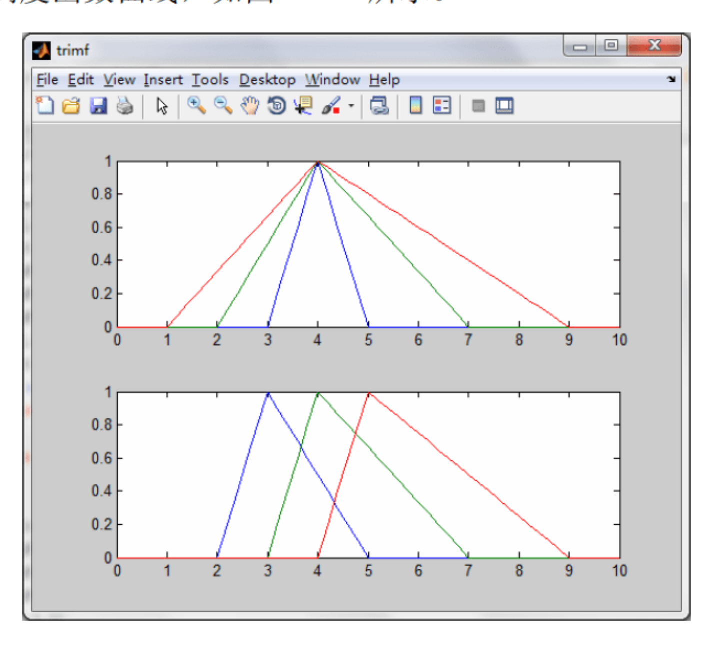


图 10-20　三角形隶属度函数曲线

（8）zmf()。建立 Z 型隶属度函数，调用格式为：

```
y = zmf(x,params)
y = zmf(x,[a,b])
```

Z 型函数是一种基于样条插值的函数，两个参数 a 和 b 分别定义样条插值的起点和终点；x 指定函数的自变量范围。

例如：

```
>>x = 0:0.1:10;
>>subplot(3,1,1);
>>plot(x,zmf(x,[2 8]));
>>subplot(3,1,2);
>>plot(x,zmf(x,[4 6]));
>>subplot(3,1,3);
```

```
>>plot(x,zmf(x,[6 4]));
>>set(gcf,'name','zmf','numbertitle','off');
```

得到 Z 型隶属度函数曲线如图 10-21 所示。

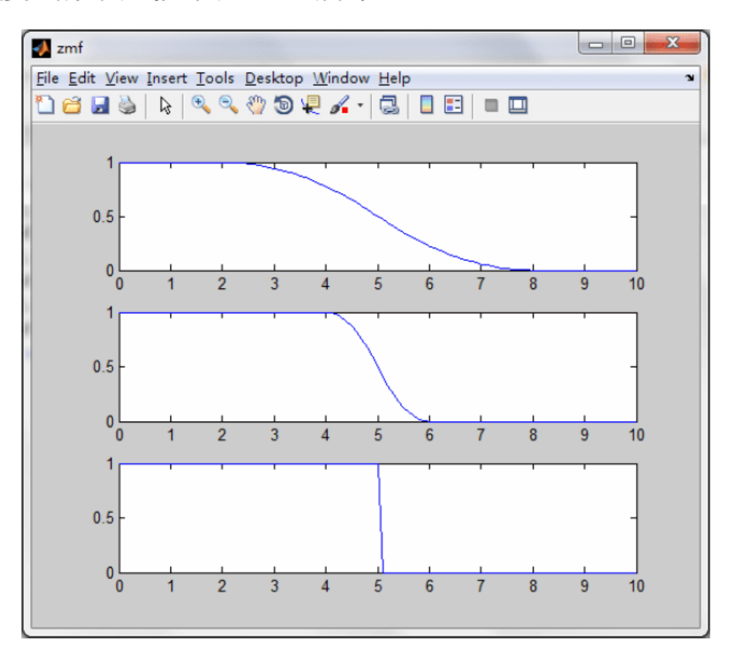


图 10-21 Z 型隶属度函数曲线

（9）psigmf()。求解两个 sigmiod 型隶属度函数之积，调用格式为：

```
y = psigmf(x,params)
y = psigmf(x,[a1 c1 a2 c2])
```

其中，参数 a1、c1 和 a2、c2 分别用于指定两个 sigmoid 型函数的形状；x 指定函数的自变量范围，函数表达式为

$$y = \frac{1}{(1+e^{-a_1(x-c_1)})(1+e^{-a_2(x-c_2)})}$$

例如：

```
>>x = (0:0.2:10)';
>>params1 = [2 3];
>>y1 = sigmf(x,params1);
>>params2 = [-5 8];
>>y2 = sigmf(x,params2);
>>y3 = psigmf(x,[params1 params2]);
>>subplot(2,1,1);
>>plot(x,y1,x,y2);
>>title('sigmf');
>>subplot(2,1,2);
>>plot(x,y3,'g-',x,y3,'o');
>>title('psigmf');
>>set(gcf,'name','psigmf','numbertitle','off');
```

得到两个 S 型函数积曲线如图 10-22 所示。

（10）dsigmf()。求解两个 sigmiod 型隶属度函数之和，调用格式为：

```
y = dsigmf(x,params)
y = dsigmf(x,[a1,c1,a2,c2])
```

该函数的用法与函数 psigmf()类似，参数 a1、c1 和 a2、c2 分别用于指定两个 Sigmoid 型函数的形状，函数表达式为

$$y = \frac{1}{(1+e^{-a_1(x-c_1)})} + \frac{1}{(1+e^{-a_2(x-c_2)})}$$

例如：

```
>>x = 0:0.1:10;
>>y = dsigmf(x,[5 2 5 7]);
>>plot(x,y)
>>xlabel('dsigmf,P=[5 2 5 7]')
```

得到两个 S 型函数和曲线如图 10-23 所示。

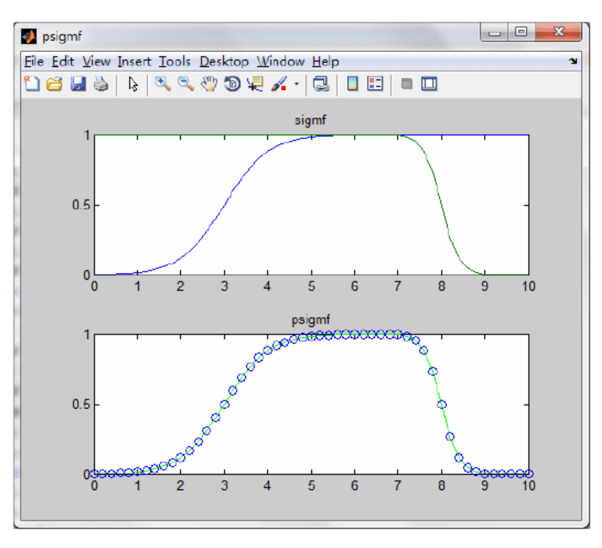


图 10-22　两个 S 型函数积曲线

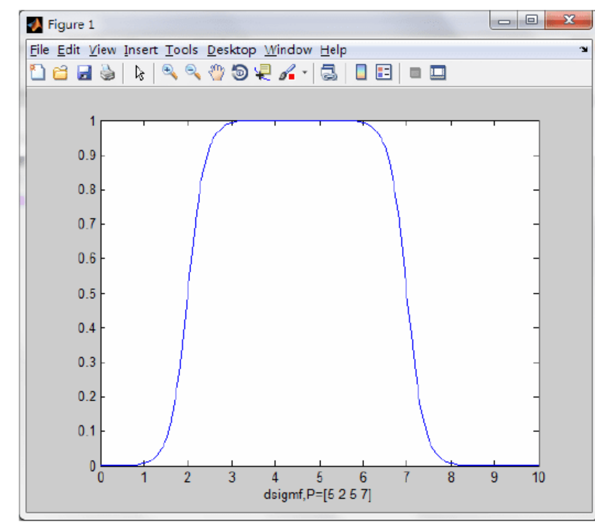


图 10-23　两个 S 型函数和曲线

（11）mf2mf()。进行不同类型隶属度函数之间的参数转换，调用格式为：

```
outParams = mf2mf(inParams,inType,outType)
```

其中，inParams 为转换前的隶属度函数的参数；outParams 为转换后的隶属度函数的参数；inType 为转换前的隶属度函数的类型；outType 为转换后的隶属度函数的类型。

该函数将尽量保持两种类型的隶属度函数曲线在形状上的近似，特别是保持隶属度等于 0.5 处的点的重合，但不可避免会丢失一些信息。因此，当再次使用该函数进行反向转换时，将无法得到与原来函数相同的参数。

例如：

```
>>x = 0:0.1:5;
>>mfp1 = [1 2 3];
>>mfp2 = mf2mf(mfp1,'gbellmf','trimf');
>>plot(x,gbellmf(x,mfp1),x,trimf(x,mfp2))
```

得到的曲线如图 10-24 所示。

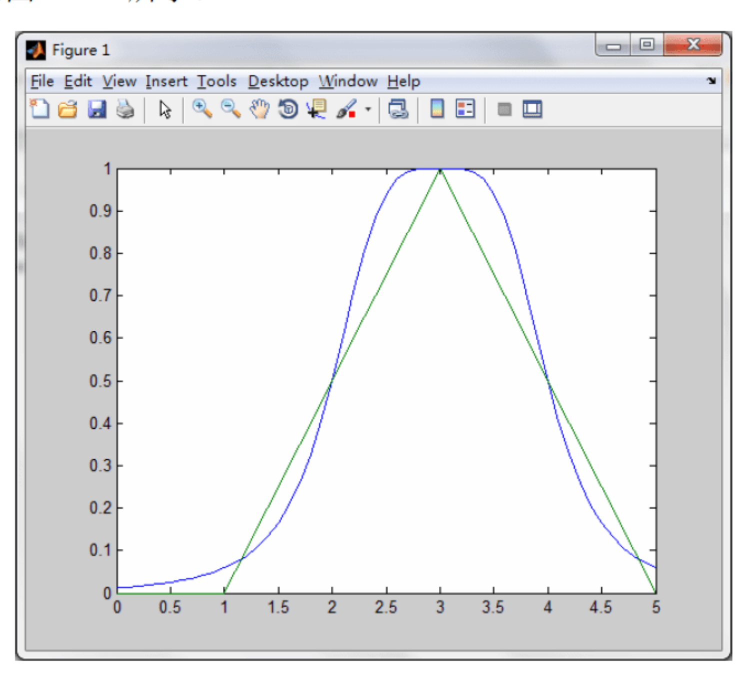


图 10-24　转换前后的隶属度函数曲线

10.4　本章小结

本章主要介绍了 MATLAB 的工具箱的使用方法。MATLAB 的每一个工具箱都是为某一学科和专业应用而特别指定的。MATLAB 的工具箱一类是功能性工具箱；另一类是领域性工具箱。功能性工具箱主要用来扩充 MATLAB 的符号计算功能、图形建模仿真功能、文字处理功能及与硬件实时交互功能，能够用于多种学科；本章主要介绍了神经网络工具箱和模糊逻辑工具箱两个应用较广的领域性工具箱。

10.5　习题

（1）MATLAB 工具箱分为几种类型，分别由哪些工具箱组成？

（2）设计一个 BP 网络，逼近以下函数：g(x)=sin(2*pi/4*x)-7，实现该非线性函数的逼近。

（3）设计一个 BP 网络，拟合非线性函数 $y = x_1^2 + x_2^2$。

（4）试用 MATLAB 模糊逻辑工具箱构建一个模糊控制系统。

第 11 章　MATLAB 外部接口

MATLAB 提供与外部交互的强大功能，MATLAB 的外部接口使得 MATLAB 可以与外部设备和程序实现数据交互和程序移植，可以扩充 MATLAB 强大的数值计算和图形显示功能，从而弥补其执行效率较低的缺点，同时增强其他应用程序进行软件开发的功能，改善了软件开发效率。通过 MATLAB 接口编程，可以充分利用现有资源，能更容易地编写出功能强大、结构简洁的应用程序。

MATLAB 具有直接对磁盘文件进行访问的功能，提供很多文件输入和输出的内建函数，它们可对二进制文件或 ASCII 文件进行方便地打开、关闭及存储等操作。

本章简要介绍 MATLAB 对磁盘文件访问的功能和 MATLAB 平台与其他平台间的外部接口，包括通过编译器生成可独立执行的代码，在 Word 和 Excel 中使用 MATLAB 资源，在 C 语言中调用 MATLAB 资源，调用外部设备及互联网资源等内容。

11.1　文本文件

MATLAB 提供的文件访问能力，可以实现数据的导入、导出，增强了其程序设计的灵活性。MATLAB 支持的文件类型包括 MATLAB 自带文件、扩展标记文件、科学数据文件、文本文件、音频文件、视频文件、表单文件及图片文件。其中，文本文件是使用最广泛且最便于阅读的文件类型，下面介绍有关文本文件的访问。

11.1.1　打开/关闭文件

1. 打开文件

根据操作系统的要求，在程序中要使用或创建一个磁盘文件时，必须向操作系统发出打开文件的命令，使用完毕后，必须关闭这个文件。

在 MATLAB 中，使用 fopen 函数打开二进制形式的文件，具体方法如下。

```
fid = fopen(filename,permission)
[fid,message] = fopen(filename,permission)
```

filename 参数用来表示要打开的文件名，文件名要带后缀。

permission 参数用来表示文件处理方式，其选项具体如下。

- **r**　以只读文件方式处理。
- **w**　以更新文件方式处理，如果文件名不存在，则生成新文件，如果文件名存在，则覆盖文件原有内容。
- **a**　以修改文件方式处理，如果文件名不存在，则生成新文件，如果文件名存在，

则在文件原有内容末尾增加新内容。

- ❑ **r+** 以读写文件方式处理读写文件，但不生成文件。
- ❑ **w+** 如果文件名不存在，则生成新文件，并可进行读写操作，如果文件名存在，则覆盖文件原有内容，并可进行读写操作。
- ❑ **a+** 如果文件名不存在，则生成新文件，并可进行读写操作，如果文件名存在，则在文件原有内容末尾增加新内容，并可进行读写操作。
- ❑ **W** 以更新文件方式处理时没有自动格式。
- ❑ **A** 以修改文件方式处理时没有自动格式。

当文件以文本形式打开时，需要在上述指定的 permission 字符（串）后加字符 t，如 rt、w+t 等。

fid 参数是由操作系统设定的一个整数，用来表示文件操作的状态及标识已打开的文件。如果返回值为–1，则表示 fopen 无法打开该文件，如以只读的形式打开不存在的文件；如果返回值为非负值，则表示文件标识。

message 参数用来表示文件操作的相关信息。

【例 11-1】 以只读方式依次打开 tan 函数、atan 函数、sin 函数、cos 函数及不存在的 sincos 函数对应文件。

在命令窗口中输入如下语句。

```
[fid1,message1]=fopen('tan.m','r')
[fid2,message2]=fopen('atan.m','r')
[fid3,message3]=fopen('sin.m','r')
[fid4,message4]=fopen('cos.m','r')
[fid5,message5]=fopen('sincos.m','r')
```

命令窗口中的输出结果如下。

```
fid1 =
     3
message1 =
        ''
fid2 =
     4
message2 =
        ''
fid3 =
     5
message3 =
        ''
fid4 =
     6
message4 =
        ''
fid5 =
    -1
message5 =
No such file or directory
```

需要说明的是，前几条语句为已存在的文件分别给出文件标识 3、4、5 和 6，这四个数字仅仅是一个标识，不同情况下运行可能数值不同。

为了后续操作的顺利进行，程序设计中每次打开文件，都要进行该操作是否正确的判断，具体如下。

```
[fid,message]=fopen(filename,'r');
if fid == -1
     disp(message);
end
```

【例 11-2】 如果某文件不存在，用函数 fopen 按只读方式打开文件。

在命令窗口中输入如下语句。

```
[fid,message] = fopen(filename,'r')
```

命令窗口中的输出结果如下所示。

```
fid =
     -1
message =
          No such file or directory
```

2. 关闭文件

在打开文件后，如果完成了对应的读写工作，应该关闭文件，这样做一是为了提高程序的可靠性，二是可以避免系统资源的浪费。在 MATLAB 中，使用与 C 语言同名的 fclose 函数关闭文件，具体方法如下。

```
status = fclose(fid)
```

其中，fid 参数即为要关闭文件的文件标识，它也是打开该文件时的返回值，如果关闭成果 status 返回值为 0，否则返回值为–1。

【例 11-3】 关闭已打开的文件。

在命令窗口中输入如下语句。

```
fid = fopen('cos.m','r')
status = fclose(fid)
```

命令窗口中的输出结果如下。

```
fid =
     11
status =
        0
```

11.1.2 二进制形式访问

对于 MATLAB，二进制文件相对比较容易处理，这些文件比较容易与 MATLAB 交互。常见的二进制文件包括.m、.dat、.txt 等。

1．读取文件

在 MATLAB 中使用 fread 函数实现从文件中读取二进制数据，并将文本内容看成一个整数序列，结果最后以矩阵的形式返回，具体使用方法如下。

```
a = fread(fid)
a = fread(fid,size)
a = fread(fid,size,precision)
```

fid 参数是打开文件时得到的文件标识。

size 参数表示读取整数的个数，它可能的形式如下所示。

- **n**　读取 n 个数据，并写入一个列向量中。
- **inf**　读取至文件末尾。
- **[m,n]**　读取 n 个数据构成列向量，存放到 m×n 矩阵中。

precision 参数表示读取的数据类型，默认情况是 uchar（即 8 位字符型），常用的数据类型如表 11-1 所示。

表 11-1　数据类型

MATLAB	描　　述
uchar	无符号字符型
schar	带符号字符型（8 位）
int8	整型（8 位）
int16	整型（16 位）
int32	整型（32 位）
int64	整型（64 位）
uint8	无符号整型（8 位）
uint16	无符号整型（16 位）
uint32	无符号整型（32 位）
uint64	无符号整型（64 位）
single	浮点数（32 位）
float32	浮点数（32 位）
double	浮点数（64 位）
float64	浮点数（64 位）

此外，还有一些类型是与平台有关的，平台不同则位数不同，如表 11-2 所示。

表 11-2　与平台有关的数据类型

MATLAB	描　　述
char	字符型
short	整型（16 位）
int	整型（32 位）
long	整型（32 位或 64 位）
ushort	无符号整型（16 位）
uint	无符号整型（32 位）
ulong	无符号整型（32 位或 64 位）
float	浮点数（32 位）

【例 11-4】 以二进制的方式读取文件 ex12_4.m，其内容如下所示。

```
function y = ex12 4(n)
y = rand(n);
```

在命令窗口中输入如下语句。

```
fclose('all');
clear
clc
fid = fopen('ex12 4.m','r');
a = fread(fid);
a1 = a'
a = fread(fid,8);
a2 = a'
fclose(fid);
```

命令窗口中的输出结果如下。

```
a1 =
  Columns 1 through 17
 102  117  110  99  116  105  111  110  32  121  32  61  32  101  120  49  50
  Columns 18 through 34
 95  52   40  110  41   13  10  121  32  61   32  114   97  110  100  40  110
  Columns 35 through 36
41    59

a2 =
     []
```

由结果可以看出，a2 是一个空矩阵，并没有得到预期结果，其原因是读语句执行后会影响文件内的控制位置。语句 “a = fread(fid)” 执行后，文件内的控制位置位于文件末尾，所以无法向下读取 8 个字节。因此每次文件打开，都应将文件内的控制位置置于开始处。

【例 11-5】 以二进制的方式读取上述文件。

在命令窗口中输入如下语句。

```
fclose('all');
clear
clc
fid = fopen('ex12 4.m','r');
a1 = fread(fid,[3,6])
a = fread(fid,inf);
a2 = a'
fclose(fid);
```

命令窗口中的输出结果如下。

```
a1 =
   102    99   111   121    32    49
   117   116   110    32   101    50
   110   105    32    61   120    95
```



```
a2 =
  Columns 1 through 17
  52   40  110  41  13  10  121  32  61  32  114  97  110  100  40  110  41
  Column 18
    59
```

由结果看出，语句“a = fread(fid,[3,6])”影响了文件内的控制位置，因此再执行语句“fread(fid,inf)”的结果维数将发生了变化。

【例 11-6】 以指定数据类型的二进制方式读取上述文件。

在命令窗口中输入如下语句。

```
fclose('all');
clear
clc
fid = fopen('ex12 4.m','r');
a = fread(fid,inf,'int8');
a1 = a'
fclose(fid);
fid = fopen('ex12_4.m','r');
a = fread(fid,inf,'int16');
a2 = a'
fclose(fid);
```

命令窗口中的输出结果如下。

```
a1 =
  Columns 1 through 17
  102  117  110  99  116  105  111  110  32  121  32  61  32  101  120  49  50
  Columns 18 through 34
  95  52  40  110  41  13  10  121  32  61   32  114   97  110  100  40   110
  Columns 35 through 36
    41   59

a2 =
  Columns 1 through 8
  30054   25454    26996    28271     31008      15648      25888       12664
  Columns 9 through 16
  24370    10292    10606     2573     8313      8253       24946       25710
  Columns 17 through 18
  28200    15145
```

可以看出，采用不同的精度将得到不同的结果。

2. 写入文件

在 MATLAB 中使用 fwrite 函数实现将二进制数据写入已打开的文件，具体使用方法如下。

```
count = fwrite(fid,a,precision)
```

【例 11-7】 将矩阵写入 ex12_7.txt 文件中。

在命令窗口中输入如下语句。

```
clear
clc
A = [1 2 3;4 5 6;7 8 9];
fid = fopen('ex12 7.txt','w');
count = fwrite(fid,A,'int32')
closestatus = fclose(fid)
```

运行结果如下。

```
count =
       9
closestatus =
             0
```

再输入如下代码。

```
clear
clc
fid = fopen('ex12 7.txt','r');
A = fread(fid,[3 3],'int32');
closestatus = fclose(fid);
B = magic(3);
C = A*B
```

运行结果如下。

```
C =
   26    38    26
   71    83    71
  116   128   116
```

11.1.3 普通形式访问

本节将介绍以普通的形式访问文件。

1. 读取文件

在 MATLAB 中使用 fgetl 函数和 fgets 函数实现将文本文件中的某一行读出，并将该行的内容以字符串形式返回，区别在于 fgetl 函数会舍弃换行符，而 fgets 函数会保留换行符，具体使用方法如下。

```
tline = fgetl(fid)
tline = fgets(fid)
```

【例 11-8】 用函数 fgetl 实现命令 type 功能。

具体代码序列如下。

```
fid = fopen('sinc.m')
while 1
    tline = fgetl(fid);
    if ~ischar(tline)
```



```
        break;
    else
        disp(tline)
    end
end
fclose(fid);
```

运行结果如下。

```
function y=sinc(x)
%SINC Sin(pi*x)/(pi*x) function.
%   SINC(X) returns a matrix whose elements are the sinc of the elements
%   of X, i.e.
%        y = sin(pi*x)/(pi*x)    if x ~= 0
%          = 1                  if x == 0
%   where x is an element of the input matrix and y is the resultant
%   output element.
%   % Example of a sinc function for a linearly spaced vector:
%   t = linspace(-5,5);
%   y = sinc(t);
%   plot(t,y);
%   xlabel('Time (sec)');ylabel('Amplitude'); title('Sinc Function')
%
%   See also SQUARE, SIN, COS, CHIRP, DIRIC, GAUSPULS, PULSTRAN, RECTPULS,
%   and TRIPULS.
%   Author(s): T. Krauss, 1-14-93
%   Copyright 1988-2004 The MathWorks, Inc.
%   $Revision: 1.7.4.1 $  $Date: 2004/08/10 02:11:27 $
i=find(x==0);
x(i)= 1;      % From LS: don't need this is /0 warning is off
y = sin(pi*x)./(pi*x);
y(i) = 1;
```

如果已知写入时的格式，可以按此格式完整读出。在 MATLAB 中使用 fscanf 函数实现已知格式文件的读取，当确定文件的 ASCII 码格式时，可用 fscanf 进行精确读取，具体方法如下。

```
a = fscanf(fid,format)
a = fscanf(fid,format,size)
[a,count] = fscanf(fid,format,size)
```

fid 参数是打开文件时得到的文件标识。size 参数与 11.1.2 节中描述类似，a 参数是读取数据构成的向量，count 参数是读取数据的个数，format 参数用于指定读入数据的格式，常用的选项如下所示。

- **%s** 按字符串进行输入转换。
- **%d** 按十进制数据进行转换。
- **%f** 按浮点数进行转换。

此外，还有其他格式，与 C 语言中 fprintf 中参数的用法相同。

在格式说明中，除了单个的空格字符可以匹配任意个数的空格字符外，通常字符在转

换时将一一匹配，函数 fscanf 将输入的文件看作一个输入流，MATLAB 根据格式来匹配输入流，并将匹配后的数据读入到 MATLAB 中。

【例 11-9】 读取文本文件(ex12_9.txt)中的数据，这些数据由 y=rand(5,6)产生。

具体代码序列如下。

```
0.8979    0.3928    0.9343    0.2101    0.9115    0.2350
0.6512    0.7548    0.3739    0.2952    0.7303    0.0671
0.2461    0.3926    0.4825    0.5320    0.8997    0.2842
0.2340    0.2654    0.2382    0.5076    0.8049    0.4317
0.5484    0.3469    0.5312    0.4991    0.4479    0.4551
```

在命令窗口中输入如下语句。

```
clear
clc
fid = fopen('ex12 9.txt','r');
d1 = fscanf(fid,'%s',[5 6])
fclose(fid);
fid = fopen('ex12 9.txt','r');
d2 = fscanf(fid,'%f',[5 6])
fclose(fid);
fid = fopen('ex12 9.txt','r');
d = fscanf(fid,'%f');
d3 = d'
fclose(fid);
```

命令窗口中的输出结果如下。

```
d1 =
    09232.00153.03743.00982.02704.09194.
    .08419.02472.01694.07432.06435.02944
    8.03012.08397.01285.02462.09143.0175
    93.01136.09530.06238.00535.07845.095
    799.05557.02062.05292.04808.04634.01

d2 =
    0.8979    0.2350    0.7303    0.5320    0.2382    0.3469
    0.3928    0.6512    0.0671    0.8997    0.5076    0.5312
    0.9343    0.7548    0.2461    0.2842    0.8049    0.4991
    0.2101    0.3739    0.3926    0.2340    0.4317    0.4479
    0.9115    0.2952    0.4825    0.2654    0.5484    0.4551

d3 =
  Columns 1 through 10
  0.8979  0.3928  0.9343  0.2101  0.9115  0.2350  0.6512  0.7548  0.3739  0.2952
  Columns 11 through 20
  0.7303  0.0671  0.2461  0.3926  0.4825  0.5320  0.8997  0.2842  0.2340  0.2654
  Columns 21 through 30
  0.2382  0.5076  0.8049  0.4317  0.5484  0.3469  0.5312  0.4991  0.4479  0.4551
```

值得注意的是，按格式读取时，必须选择正确的格式才能正确地复现数据。

2. 写入文件

在 MATLAB 中使用 fprintf 函数实现将数据按给定格式写入文件，具体方法如下。

```
count = fprintf(fid,format,y)
```

fid 参数是打开文件时得到的文件标识，y 参数用于指定要写入的数据，count 参数用于返回成功写入的字节数，format 参数用于指定写入文件的数据格式，常用的格式如下。

- ❑ **%e** 科学计数形式，即数值表示成形式。
- ❑ **%f** 固定小数点位置的数据形式。
- ❑ **%g** 在上述两种格式中自动选择较短的格式。

还可以包括数据占用的最小宽度和数据精度的说明。可同时使用\n、\r、\t、\b 等分别代表换行、回车、Tab、退格等字符，用\\代表反斜线\，%%代表百分号%。

【例 11-10】 向文件 ex12_10.dat 中写入数据。

在命令窗口中输入如下语句。

```
clear
clc
y = rand(5)
fid = fopen('ex12_10.dat','w');
fprintf(fid,'%6.3f',y);
fclose(fid);
clear
fid = fopen('ex12_10.dat','r');
ey = fscanf(fid,'%f');
ey1 = ey'
fclose(fid);
fid = fopen('ex12_10.dat','r');
ey2 = fscanf(fid,'%f',[4 4])
fclose(fid);
```

命令窗口中的输出结果如下。

```
y =
    0.9794    0.3992    0.6202    0.6828    0.3676
    0.9269    0.9429    0.4357    0.1043    0.4220
    0.5976    0.2143    0.4158    0.0835    0.5757
    0.5837    0.1705    0.2852    0.0134    0.1860
    0.1131    0.3469    0.9051    0.9188    0.0222


ey1 =
  Columns 1 through 10
  0.9790  0.9270  0.5980  0.5840  0.1130  0.3990  0.9430  0.2140  0.1700  0.3470
  Columns 11 through 20
  0.6200  0.4360  0.4160  0.2850  0.9050  0.6830  0.1040  0.0830  0.0130  0.9190
  Columns 21 through 25
  0.3680  0.4220  0.5760  0.1860  0.0220


ey2 =
    0.9790    0.1130    0.1700    0.4160
```

```
    0.9270    0.3990    0.3470    0.2850
    0.5980    0.9430    0.6200    0.9050
    0.5840    0.2140    0.4360    0.6830
```

本例中，“%6.3f”表示占 6 个字符位，小数点后的精度是 3 位，MATLAB 默认设置是小数点后 4 位。

11.1.4 文件内的位置控制

打开文件读写数据时，需要判断和控制文件的读写位置，需要读写指定位置上的数据等。在读写文件时，MATLAB 会自动创建一个文件位置指针来管理维护文件读写数据的起始位置。

读写数据时，系统默认的方式是从文件头顺序向后读写数据至文件尾。操作系统通过文件指针来指示当前的文件位置，通过控制指针实现文件内的位置控制。MATLAB 提供如表 11-3 所示的位置控制函数。

表 11-3 文件位置控制函数

函数	功能
feof	判断指针是否在文件尾
fseek	设定文件指针位置
ftell	获取文件指针位置
frewind	设置指针至文件头位置

下面将分别介绍表中函数。

（1）feof 函数的具体用法如下。

```
status = feof(fid)
```

fid 参数是打开文件时得到的文件标识。status 参数为 1 表示指针在文件尾，否则为 0。

（2）fseek 函数的具体用法如下。

```
status = fseek(fid,offset,origin)
```

fid 参数是打开文件时得到的标识。offset 参数是偏移量，以字节数为单位（正整数表示往文件尾方向移动指针，0 表示不移动指针，负整数表示往文件头方向移动指针）。origin 参数是基准点（“bof”和–1 表示文件头位置，“cof”和 0 表示目前位置，“eof”和 1 表示文件尾位置）。status 参数为 0 表示操作成功，否则为 1。

（3）ftell 函数的具体用法如下。

```
position = ftell(fid)
```

fid 参数是打开文件时得到的文件标识。position 参数表示距离文件头位置的字节数，如果为–1 表示操作失败。

（4）frewind 函数的具体用法如下。

```
frewind(fid)
```

fid 参数是打开文件时得到的文件标识。

【例 11-11】 利用文件内的位置控制读取文件。

在命令窗口中输入如下语句。

```
clc
clear
fid = fopen('magic.m','r');
p1 = ftell(fid)
a1 = fread(fid,[4 4])
status = fseek(fid,10,'cof');
p2 = ftell(fid)
a2 = fread(fid,[4 4])
frewind(fid);
p3 = ftell(fid)
a3 = fread(fid,[4 4])
status = fseek(fid,0,'eof');
p4 = ftell(fid)
d = feof(fid)
fclose(fid);
```

命令窗口中的输出结果如下。

```
p1 =
     0

a1 =
   102   116    32    32
   117   105    77   109
   110   111    32    97
    99   110    61   103

p2 =
    26

a2 =
    73    77    99   117
    67    97    32    97
    32   103   115   114
    32   105   113   101

p3 =
     0

a3 =
   102   116    32    32
   117   105    77   109
   110   111    32    97
    99   110    61   103

p4 =
     1043
```

```
d =
     0
```

由本例结果可以看出，文件头位置是 0，文件经过读取指针会相应地移动，文件指针受到指定函数的控制。

【例 11-12】 通过演示数据文件指定位置数据读取的方法。

在命令窗口中输入如下语句。

```
A = magic(4);
fid = fopen('c:\data.txt','w');
fprintf(fid,'%d\n','int8',A);
fclose(fid);
fid = fopen('c:\data.txt','r');
frewind(fid);
if feof(fid)==0
    [B,COUNT] = fscanf(fid,'%d\n')
    position = ftell(fid)
end
if feof(fid)==1
    status = fseek(fid,-4,'cof')
    [C,COUNT] = fscanf(fid,'%d\n')
end
fclose(fid);
```

命令窗口中的输出结果如下。

```
B =
   105
   110
   116
    56
    16
     5
     9
     4
     2
    11
     7
    14
     3
    10
     6
    15
    13
     8
    12
     1

COUNT =
    20
```

```
position =
    54
status =
     0
C =
     2
     1
COUNT =
     2
```

11.2 MATLAB 与 Word 混合使用

Notebook（笔记本）是 MATLAB 与 Microsoft Word 的结合使得用户可以在 Word 环境下灵活使用 MATLAB 的数学运算与可视化功能，为用户提供了融文字处理、工程设计及科学计算为一体的工作环境。Notebook 将文字编辑与 MATLAB 计算命令的实时演示相结合，可用于论文、著作、讲义、教材及科技报告等方面，使得文稿做到了动静结合、图文并茂，且可以随时验证运算的正确性，所生成的 M-book 文件可以与 PowerPoint、Authorware 等应用软件连接起来交互使用。

11.2.1 Notebook 的安装

1．安装

以 MATLAB 2010 版为例，Notebook 的安装过程独立于 MATLAB 的安装过程，在安装了 Word 和 MATLAB 后再进行 Notebook 的安装，具体操作步骤如下。

（1）启动 MATLAB，进入 MATLAB 工作环境。

（2）在 MATLAB 命令窗口中运行“notebook -setup”命令，出现如下提示。

```
>> notebook -setup
```

显示如下结果。

```
Welcome to the utility for setting up the MATLAB Notebook
for interfacing MATLAB to Microsoft Word
Setup complete
```

若安装程序无法找到所需要的文件，也会提示用户指定“winword.exe”和“normal.exe”文件的路径。

2．创建 M-book 文件

安装完后，可以启动 Notebook，也就是创建 M-book 文件。

创建 M-book 文件有两种方法：可以从 MATLAB 命令窗口启动，也可以从 Word 中启动。

（1）从 MATLAB 中启动 Notebook，在 MATLAB 命令窗口输入“notebook”命令就可以启动 Notebook，具体语法如下。

```
notebook                      %打开一个新的 M-book 文档
notebook FileName             %打开已存在的 M-book 文档
```

其中，FileName 应包含文件的完整路径及文件名。在命令窗口中打开已经建立的 M-book 文件的命令如下。

```
notebook 'C:\My Documents\MyMbook0801.doc'
```

执行完此命令则会出现新的“M-book”文档式样的 Word 窗口。

（2）从 Word 中也可以直接启动 notebook。当以默认的方式打开 Word 时，所得到的 Word 窗口是按默认的模板形式创建的。创建 M-book 文件的步骤如下。

选择 Word 窗口的菜单项【文件】/【新建】命令，在出现的对话框中找到并选择“m-book.dot”图标，在该“新建”对话框中单击“确定”按钮，如图 11-1 所示。若此时 MATLAB 尚未启动，则 MATLAB 自动启动，出现新的“m-book”文档样式的 Word 窗口。

与普通的 Word 界面相比，m-book 的界面多了一个“Notebook”菜单。Notebook 菜单如图 11-2 所示，其菜单项功能如表 11-4 所示。此外，用户还可以选择“New M-book”菜单项新建 M-book 文档，选择“About MATLAB Notebook”菜单项查看版本信息。

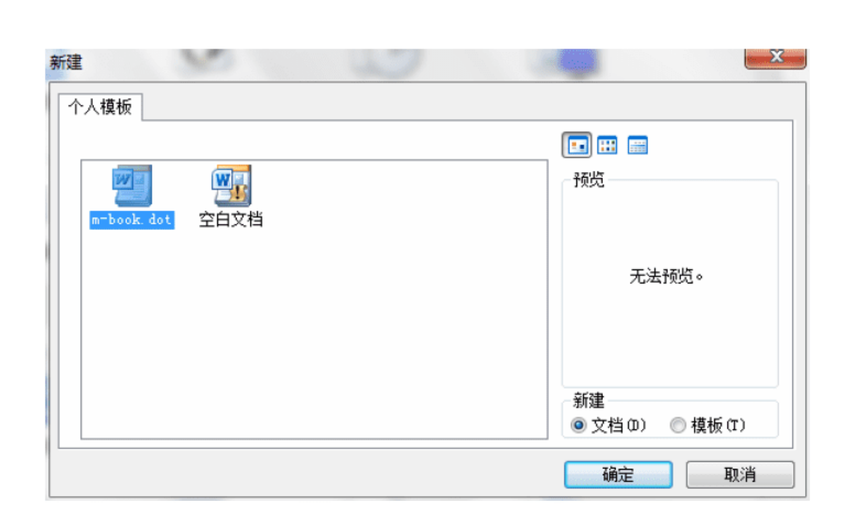


图 11-1　新建对话框

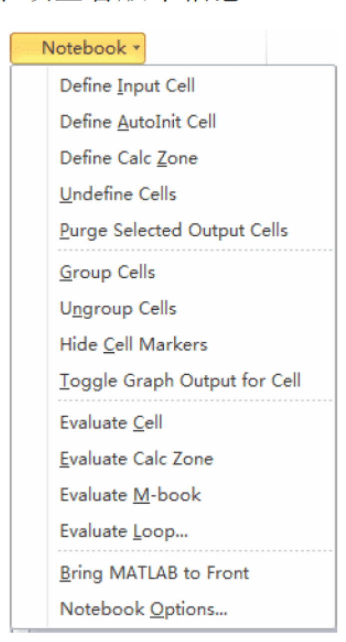


图 11-2　Notebook 菜单

表 11-4　Notebook 菜单项功能表

下 拉 菜 单	组 合 键	功 能
Define Input Cell	Alt+I	定义输入单元
Define AutoInit Cell	Alt+A	定义自动初始化单元
Define Calc Zone	Alt+Z	定义计算区
Undefine Cells	Alt+U	将单元转换为文本
Purge Output Cells	Alt+P	清除输出单元
Group Cells	Alt+G	定义单元组
Ungroup Cells	Alt+p	将单元组转换为单个单元
Hide/Show Cell Markers	Alt+C	隐藏/显示单元标志
Toggle Graph Output for Cell		为每个单元锁定图形输出

续表

下拉菜单	组合键	功能
Evaluate Cell	Ctrl+Enter	运行当前单元或单元组
Evaluate Calc Zone	Alt+Enter	运行当前计算区
Evaluate M-book	Alt+M	运行 M-book 中所有单元
Evaluate Loop	Alt+L	循环运行单元
Bring MATLAB to front		将 MATLAB 置于屏幕之前
Notebook Options⋯	Alt+O	定义输出显示选项

11.2.2 Notebook 的使用

Notebook 具有 Word 的全部功能。在 M-book 模板中，图像、文档、公式、编辑、排版等与 Normal 模板中完全相同。下面介绍如何将运算命令传递给 MATLAB 并将运算结果显示出来的方法。

在 Notebook 中，在 Word 和 MATLAB 之间参与信息交换的部分被称为单元或单元组（Cells or Cell group）。M-book 向 MATLAB 传递命令，称为输入单元（Input cells）；如果输入单元需要在启动 M-book 文件时自动进行初始化计算，则称之为自动初始化单元（AutoInit Cell）；由 MATLAB 返回给 M-book 的计算结果则称之为输出单元（Output cells）.

Notebook 对常用的 MATLAB 函数、命令及各单元格式的默认设置如表 11-5 所示。

表 11-5　Notebook 对常用各单元格式的默认设置

样式名	含义	字体
Input	输入单元	10 磅（points）深绿色英文粗体 Courier New
Output	输出单元（数据和字符）	10 磅（points）蓝色英文细体 Courier New
AutoInit	自动初始化单元	10 磅（points）深蓝色英文粗体 Courier New
Error	出错提示	10 磅（points）深红色英文粗体 Courier New

用户可通过菜单项【格式】/【样式】，修改 Notebook 模板的样式或添加新样式。

1. 输入单元

在 MATLAB 中合法的命令、注释都可以定义为输入单元，输入单元的创建有两种方法。

（1）创建不运行输入单元。首先，以文本方式输入 MATLAB 命令，然后用光标选中该命令，按快捷键 Alt+D 或选取菜单【Notebook】/【Define Input Cell】命令，则被选中文本变为输入单元。

（2）创建并同时运行输入单元。以文本输入方式输入 MATLAB 命令，用光标选中该命令，按快捷键 Ctrl+Enter 或选择菜单项【Notebook】/【Evaluate Cell】，则被选中的文本自动变成输入单元，应得出运算结果，即输出单元。

【例 11-13】 在 M-book 中创建输入单元。

在文本中输入文字“生成向量，如”，用光标高亮选中“”文本，按快捷键 Alt+D，则创建了输入单元，显示如下。

```
生成向量，如 x=1:0.1:3;
```

【例 11-14】 创建并运行输入单元。

在文本中输入 MATLAB 命令“y=sin(x)”，用光标高亮选中“y=sin(x)”文本，按快捷键 Ctrl+Enter，或选择菜单【Notebook】/【Evaluate Cell】命令，则所选中的文本形式命令就会自动变成输入单元并得到运算结果，即输出单元，显示结果如下。

```
y =sinx
y =
  Columns 1 through 10
  0.8415  0.8912  0.9320  0.9636  0.9854  0.9975  0.9996  0.9917  0.9738  0.9463
  Column 11
    0.9093
```

其中变量 x 的值来自于 MATLAB 空间，M-book 文件与 MATLAB 命令窗口共享同一个工作空间。若语句后加分号，则不会显示出运行结果，注意不要将中文标点混杂在 MATLAB 命令中。

2. 自动初始化单元

自动初始化单元与输入单元功能唯一的不同之处在于：当用户启动一个 M-book 文件时，包含在其中的自动初始化单元会自动被送去运算。若用户需要在打开文件时初始化 MATLAB 工作内存，则自动初始化单元是很好的选择。

创建自动初始化单元的方法是用光标高亮选中文本形式的 MATLAB 命令或已存在的输入单元，然后选择菜单项【Notebook】/【Define AutoInit Cell】，则选中的文本形式的 MATLAB 命令自动变为 AutoInit 格式。新定义的自动初始化单元并不自动运算，需要使用与输入单元同样的方法。

3. 单元组

单元组（Cell group）是由多行输入的单元或自动初始化的单元所组成的一个整体。使用单元组可以保证 MATLAB 命令结构（如循环结构等）、输出结果和图形的完整性。

创建单元组的具体方法如下。

（1）选中多行文本形式的 MATLAB 命令，依次选择菜单【Notebook】/【Define Input Cell】或【Notebook】/【Define AutoInit Cell】命令。

（2）选中多行独立的输入单元或自动初始化单元，依次选择菜单【Notebook】/【Group Cells】命令，则生成了一个独立单元性质的单元组。

【例 11-15】 创建单元组。

在 M-book 中输入以下文本。

利用“:”生成向量，例如，x=1:0.1:3;

```
y=sin(x)
```

画出正弦波形：

```
plot(x,y)
```

选择所有文本行，使用菜单【Notebook】/【Group Cells】命令创建单元组，可以看到“画出正弦波形”的文本内容移到单元组之后。

使用单元组时，有如下几点说明。

若选中单元组的多行文本有独立成行的文本，则创建单元组的同时会将中间的文本内容移至单元组后。

若选择了菜单项【Notebook】/【Show Cell Markers】，则会出现整体标志“[ ]”，而如果是单独输入的单元，则“[ ]”标志在该行的首尾，如果是单元组则“[ ]”在单元组的首尾。

使用循环、分支结构生成完整图形的命令行时，必须使用单元组，若用单独的输入单元，则运行时显示出错警告。

如果要将单元组恢复为单独的单元，选择菜单【Notebook】/【Ungroup Cells】命令。

单元组运行时，无论输入单元中显示运算结果的命令次序如何，数据结果总是显示在图形结果前。

4. 输出单元

输出单元包含 MATLAB 的输出结果，包括数据、图形和出错信息。输出单元是输入单元或单元组运算后产生的，总是紧跟在输入单元或单元组后。若输入单元修改后重新运行，则新的输出单元替换原有的输入单元。

【例 11-16】运行单元组查看输出单元。

选中上例中创建的单元组，按快捷键 Ctrl+Enter，则出现输出单元如下。

利用“：”生成向量，例如，x=1:0.1:3;

```
y=sin(x)
plot(x,y)
y =
Columns 1 through 10
0.8415  0.8912  0.9320  0.9636  0.9854  0.9975  0.9996  0.9917  0.9738  0.9463
Columns 11 through 20
0.9093  0.8632  0.8085  0.7457  0.6755  0.5985  0.5155  0.4274  0.3350  0.2392
Column 21
0.1411
```

画出正弦波形，如图 11-3 所示。

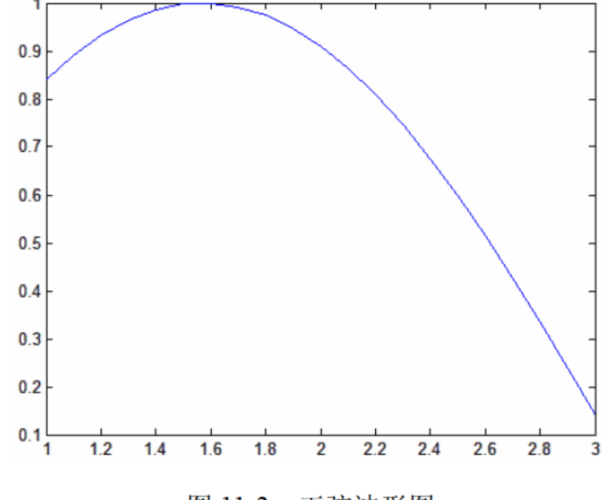


图 11-3　正弦波形图

输出单元为 y 数值和图形，输出单元也用“[ ]”标志，输入单元组的运行结果也是输出单元组。

5. 计算区

计算区（Calc Zone）是一个由文本、输入单元及输出单元组成的连续区，用于实现某个具体的问题。在计算区中，用户不受计算区外格式定义的限制，可根据需要安排段落、标题、格式及分栏等。

创建计算区的方法是先选定包含输入单元、输出单元及文本的一个连续区，选择菜单【Notebook】/【Define Calc Zone】命令；若要运行该计算区，可将光标置于计算区的意义位置，选择菜单【Notebook】/【Evaluate Calc Zone】命令。

创建计算区后，计算区定义为 Word 中的 1 个节，在计算区的首尾会出现分隔符。

6. 取消定义单元

如需要对已定义的单元和单元组进行取消，将其恢复为不会再运行的普通文本，具体方法是选择选定单元，选择菜单【Notebook】/【Undefined Cells】命令，或当光标置于单元中时单击 Alt+U，则单元恢复为“Normal”样式的文本。若输入单元或单元组被转换为文本，输出单元也转换为文本。

11.2.3 Notebook 的实际应用

Notebook 中 MATLAB 的使用与一般情况下 MATLAB 的用法有所不同。

1. 内存初始化

M-book 文件的所有计算变量都放在 MATLAB 的工作空间中，工作空间中的变量是由各 M-book 文件和 MATLAB 命令窗口共同产生的。

当用户同时打开几个 M-book 文件，或在 MATLAB 命令窗口与 M-book 文件间进行交互时，要注意各文件中变量的相互影响。为了避免其他文件或命令窗口中变量的改变影响该文件，保证文件输入/输出数据的一致性，可以用“Clear”命令作为该文件的第一个自动初始化单元。

2. M-book 文件的运行

选择菜单【Notebook】/【Evaluate M-book】命令运行整个 M-book 文件，即文档中的所有输入单元均被送至 MATLAB 中运行，可保证整个 M-book 文件中的所有命令、数据及图形的一致性。

不论光标在文档中的何处，运行总是从文件的首部开始。在整个 M-book 文件运行时，不但会把所有原输出单元中的内容刷新，还会补写新的输出单元。但“Evaluate M-book”命令可能会造成排版混乱，由于 M-book 模板的输出单元是“两端对齐”，对于图形通常是“居中”，则会引起版面混乱，尤其是对较大的 M-book 文件。

3. 输出单元的格式控制

输出单元包括数据、图像及错误信息，通过“Notebook Options”设置输出数据。具体方法是选择菜单【Notebook】/【Notebook Options…】命令，出现如图 11-4 所示的对话框。

（1）输出数据的格式控制。图中的“Numeric format”选项栏中，可以设置输出数据的格式，共有“Long”、“Hex”、“Bank”、“Plus”、“Short”和“Rational”六种，其作用与命

令窗口中的 format 命令一样，可通过下拉列表进行设置，也可在 M-book 中使用 format 命令对输入单元进行设置。

（2）输出数据间的空行控制。使用对话框中的“Loose”和“Compact”单选按钮可控制输入/输出单元间的空白区间。选择“Loose”，则 M-book 文档的输入单元与输出单元间就加入一个空行；选择“Compact”，则输入单元与输出单元、输出单元与输出单元之间均没有空行。

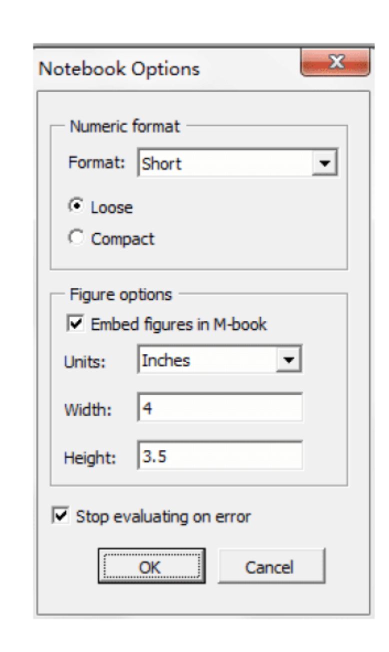


图 11-4　Notebook Options 对话框

（3）图形嵌入控制。图中“Figure Options”框架中的内容用来设置输出图形控制的。“Embed Figures in M-book”复选框用来决定输入单元中的绘图命令是否向 M-book 文档输出图形。M-book 的默认设置为“选中”，表示输出图形有可能被嵌入至 M-book 文档中；若处于“未选中”状态则 M-book 文档中没有输出图形，图形将输出到另外的图形窗口中。一旦图形被嵌入至 M-book 中，则与 Word 中图形一样，可被移动、裁剪、编辑和缩放。

（4）嵌入图形的大小控制。通过设置“Figure options”中的“Units”、“Width”和“Height”栏目，可以控制嵌入图形的大小。值得注意的是，当制定了嵌入图形的大小，则 MATLAB 中控制宽高比的 axis 命令就不能发挥原有的作用。

（5）嵌入图形的打印输出控制。为了使彩色打印机有正确的彩图输出或黑白打印机有正确的“灰度”图形输出，应选中“Use 16-Color Figures”复选项，否则打印出的面、块图将是一片黑色。

（6）嵌入图形背景控制。默认情况下，嵌入 Word 的图形背景色应该是“灰/白”的。若所嵌入的图形出现“灰/黑”背景色，则可以采取如下两种方法。

① 打开“Notebook Option”对话框，选中“Embed Figures in M-book”复选项，单击“OK”按钮，重新运行输入单元。

② 在 MATLAB 命令窗口中，运行

```
>>whitebg('white')
```

或运行以下命令。

```
>>close;
>>colordef white
```

再重新运行输入单元。

【例 11-17】 绘制三维 peaks 函数图形。

选择菜单【Notebook】/【Notebook Options...】命令，将“Width”和“Height”均设置为 2，输入如下文本，用“Evaluate M-book”命令运行该文本，则 M-book 中显示三维 peaks 函数图形，如图 11-5 所示。

绘制 peaks 函数的三维曲面图[x,y,z]=peaks;

```
mesh(x,y,z)
```

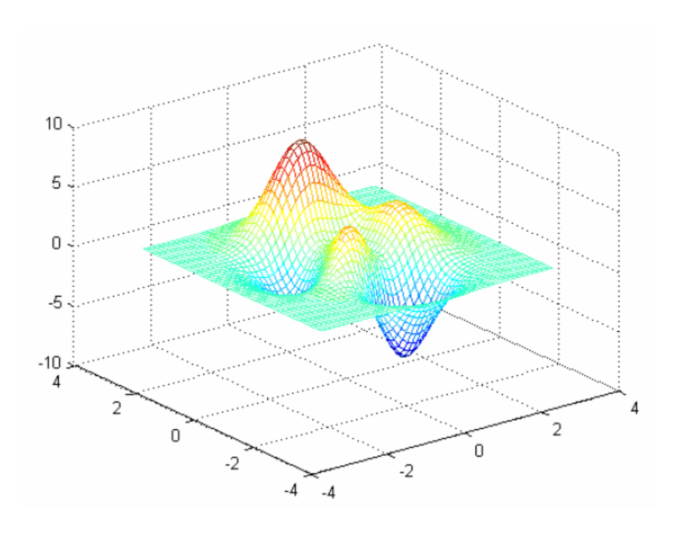


图 11-5　三维曲面图

4．单元的循环控制

Notebook 提供了循环运行单元的命令，预先选定需要循环运行的输入单元，然后选择菜单【Notebook】/【Evaluate Loop】命令，则出现“循环运行”对话框，如图 11-6 所示。

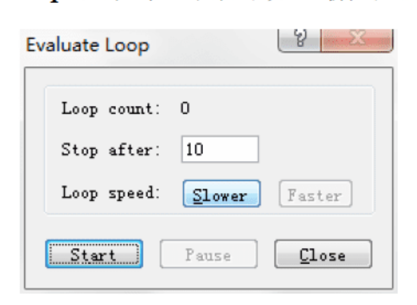


图 11-6　Evaluate Loop 对话框

可以在对话框的“Stop After”栏输入重复运行的次数，“Start”按钮用于开始运行，“Slower”按钮用于在每次循环后加入延迟，“Pause”按钮用于暂停运行。

5．删除 M-book 文件所有选中的输出单元

如果需要删除所有输出单元，可选择菜单【Notebook】/【Purge Selected Output Cells】命令，则会删除所有输出单元。

11.3　MATLAB 与 Excel 混合使用

本章主要介绍如何在 Excel 平台下使用 MATLAB 的资源。

MATLAB 作为功能强大的数学软件，数据计算和图像显示是优势，而微软的 Excel 同样具有较强的数据统计和显示能力。Excel 在一些较为简单的图像显示上方便易用，对一些复杂的图像显示就差强人意了。MATLAB 提供了两种方法实现了与 Excel 的数据共享和功能交互，使得这两大软件有机地结合起来。

MATLAB 提供了 Spreadsheet Link 将 Microsoft Excel 和 MATLAB 完美结合，即在 Excel 表单中利用 MATLAB 资源，包括科学计算和绘图功能。它的工作原理是，首先在 Excel

表单中创建命令，其次传递给 MATLAB 进行后台处理，最后将后台处理结果回传到 Excel 表单中。

Excel Link 的运行机制如图 11-7 所示。

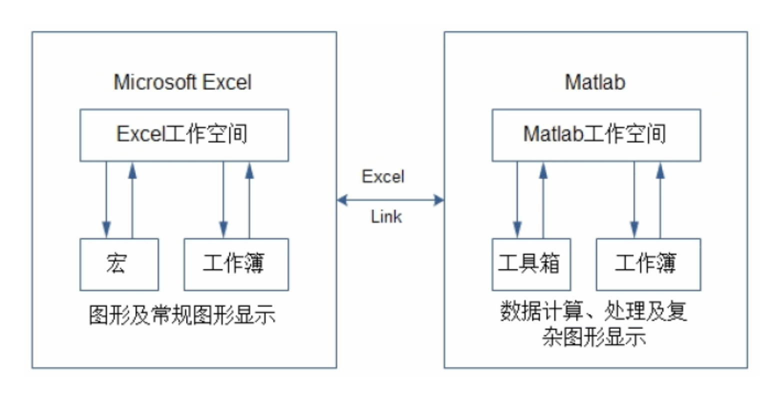


图 11-7　Excel Link 的运行机制

11.3.1　Spreadsheet Link 的安装

Excel Link 是一个插件软件，在基于 Windows 的环境中集成了 Excel 和 MATLAB。Excel 和 MATLAB 的协作可以在 Excel 工作表和宏工具中使用 MATLAB 的数值分析、数学计算和绘图功能。Excel Link 保持了两者之间数据的同步交换，它的工作环境必须是微软的 Windows，如 Windows XP 等，系统必须安装 Excel 和 MATLAB，而且安装 MATLAB 时选择安装 Excel Link。

上述系统要求具备后，还需对 Excel Link 进行配置，以实现 MATLAB 与 Excel 的连接。

在 MATLAB 中使用 Spreadsheet Link 的前提是已经安装 Microsoft Excel 系列产品中的一个，下面给出 Spreadsheet Link 的安装步骤。

（1）启动 Microsoft Excel，选择【文件】/【选项】/【加载项】/【转到】命令，如图 11-8 所示。

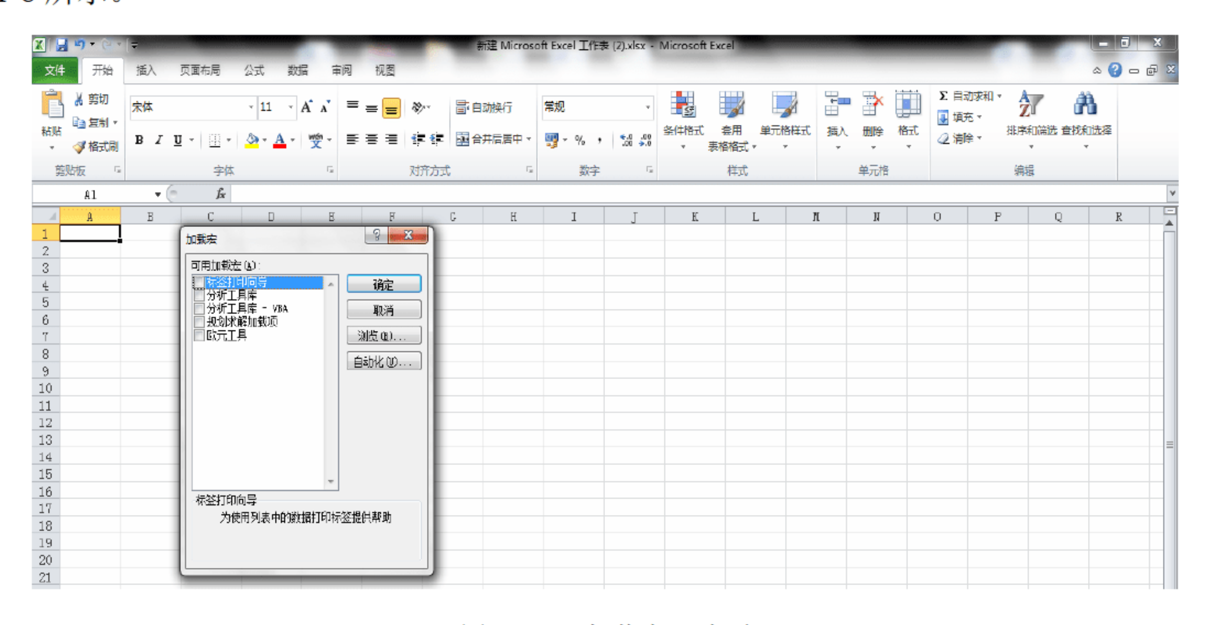


图 11-8　“加载宏”选项

（2）在“加载宏”对话框中单击“浏览”按钮，选择 MATLAB 路径下的\toolbox\exlink 子目录中的 excllink.xla 文件，然后单击“确定”按钮，如图 11-9 所示。

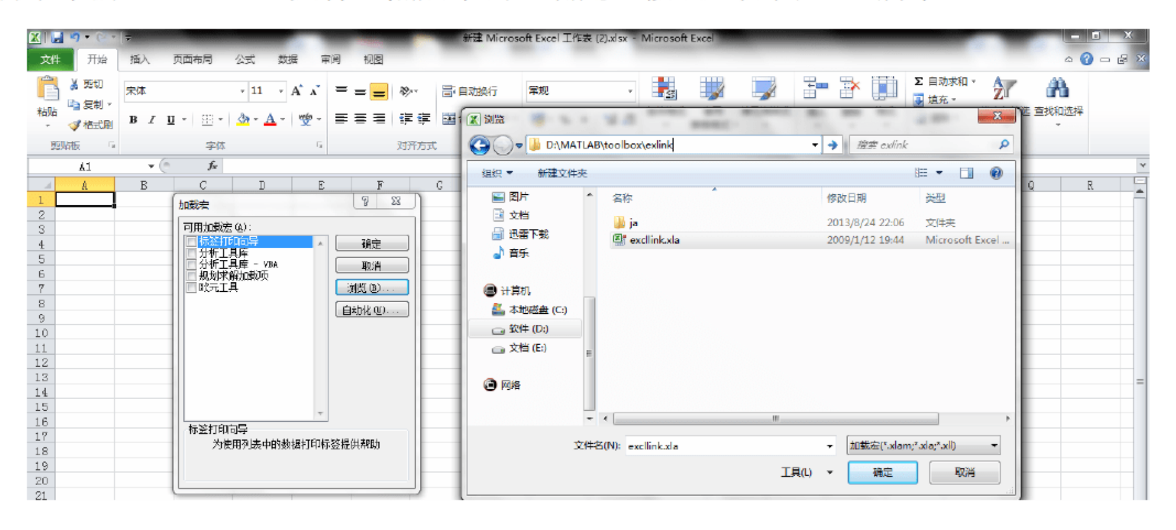


图 11-9 加载宏文件

（3）返回“加载宏”对话框，此时已经添加并选中“Spreadsheet Link EX 3.0.3 for use with MATLAB”选项，如图 11-10 所示。单击“确定”按钮即可加载 MATLAB，并出现如图 11-11 所示的 Excel 窗口。



图 11-10 已经选中“Spreadsheet Link EX 3.1.1 for use with MATLAB”选项

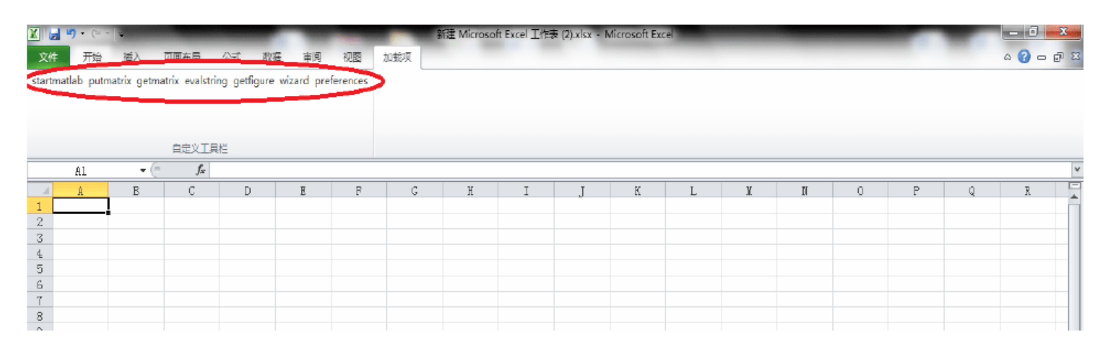


图 11-11 Excel 窗口

其中，增加了 Spreadsheet Link 的工具条，如图 11-12 所示。



图 11-12　Spreadsheet Link 工具条

Spreadsheet Link 工具条中出现了 7 个执行 MATLAB 的命令按钮，它们的含义如下。

StartMATLAB：启动 MATLAB。

putmatrix：把数据传给 MATLAB。

Getmatrix：从 MATLAB 提取数据。

Evalstring：执行 MATLAB 命令。

Getfigure：获取当前的 MATLAB 图形。

Wizard：打开 MATLAB 函数向导。

Preferences：打开 MATLAB/Spreadsheet Link EX 参数设置对话框。

Spreadsheet Link 工具条在不需要时，可以隐藏起来，方法是在工具条上右击，在弹出的菜单中取消选择“Spreadsheet Link EX”选项。

11.3.2　Spreadsheet Link 的启动和退出

按照上述步骤安装 Spreadsheet Link 后，将在每次启动 Excel 时自动启动 Spreadsheet Link 和 MATLAB。

如果希望改变此种启动方式，可以在 Excel 编辑框中输入“=MLAutoStart("no")语句，执行后即可改变设置，如图 11-13 所示；当然如果希望恢复原设置，可以输入=MLAutoStart("yes")语句。

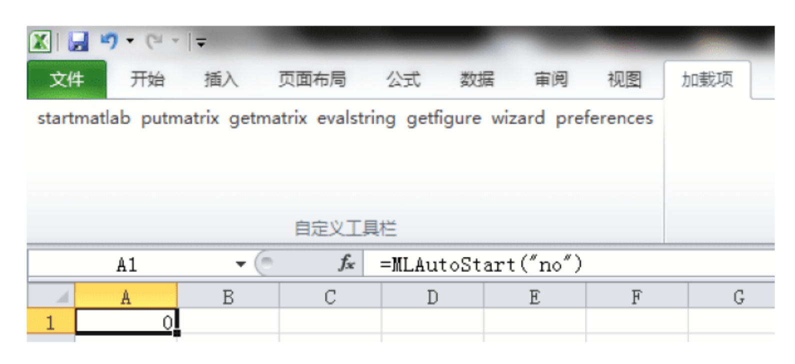


图 11-13　输入命令

对于 Spreadsheet Link 和 MATLAB 没有自动启动的情况，可以在 Excel 平台中手动启动。

首先在“工具”菜单中选择“宏”选项，如图 11-14 所示。

打开如图 11-15 所示的“宏”对话框，输入“MATLABinit”，单击“执行”按钮后即可启动 Spreadsheet Link 和 MATLAB。

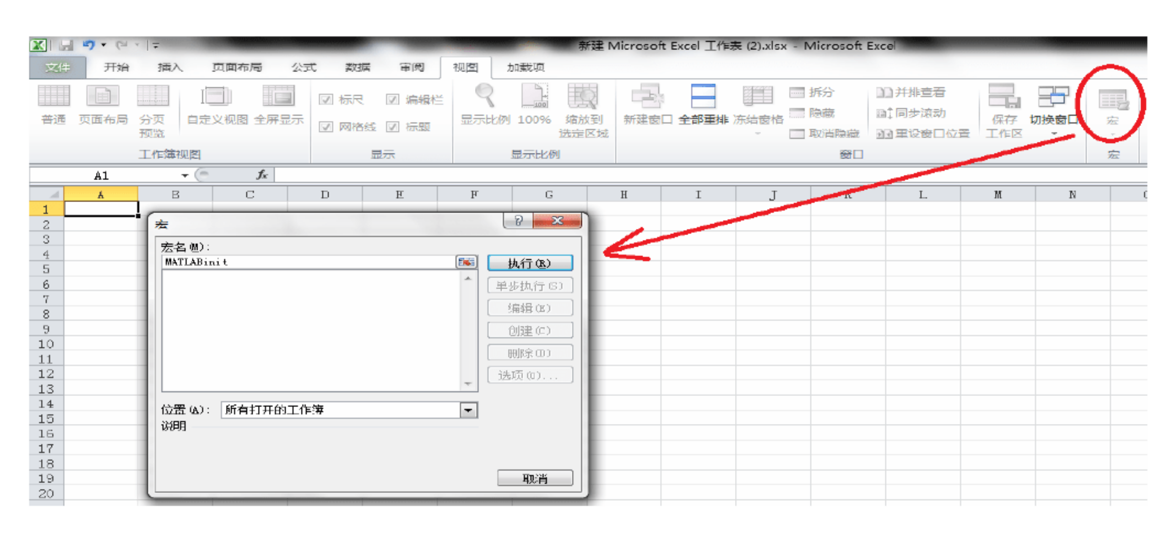


图 11-14　选择“宏”选项

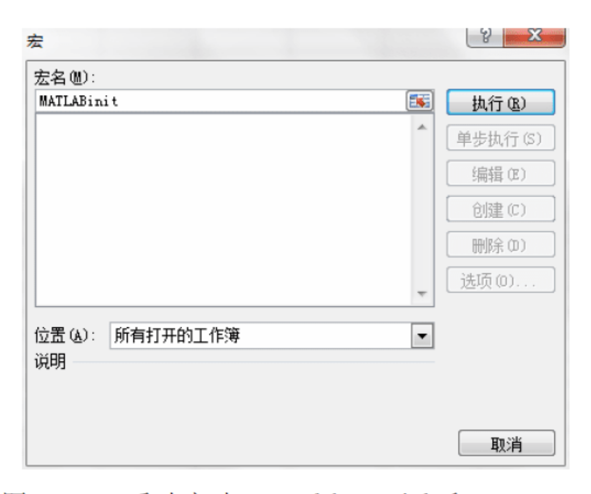


图 11-15　手动启动 Spreadsheet Link 和 MATLAB

当退出 Excel 时将自动退出 Spreadsheet Link 和 MATLAB。如果希望在 Excel 平台中退出 Spreadsheet Link 和 MATLAB，只需在编辑框中输入“=MLClose()”语句即可，如图 11-16 所示。

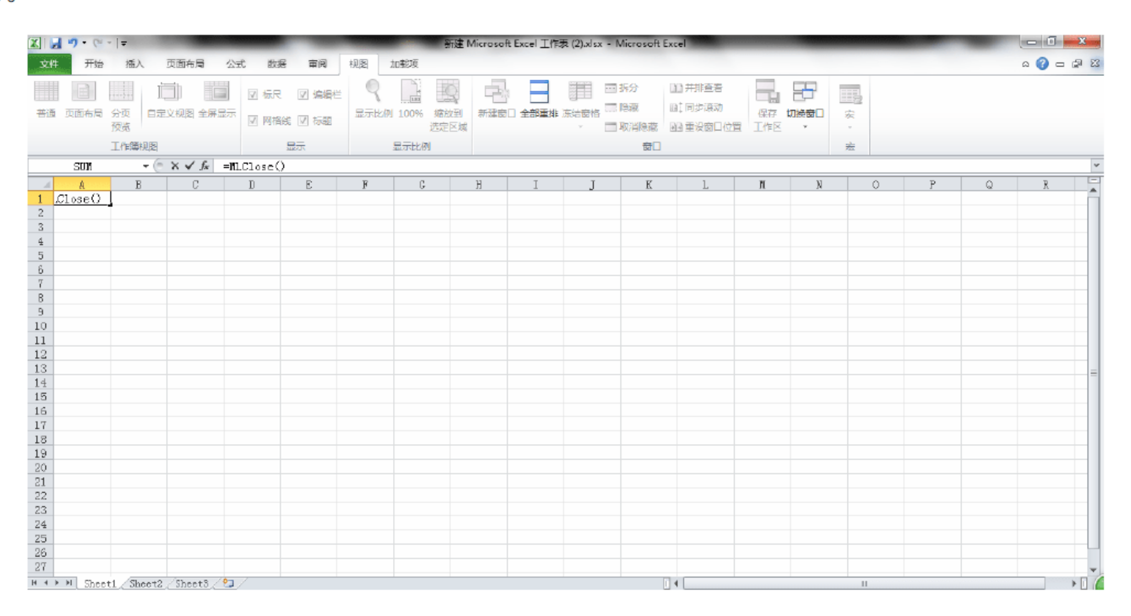


图 11-16　手动退出 Spreadsheet Link 和 MATLAB

11.3.3　Spreadsheet Link 的实际应用

在应用 Spreadsheet Link 时，主要是实现 Excel 数据的读入，MATLAB 对数据的处理和显示，以及将处理结果显示在 Excel 中。下面通过一个实例说明 Spreadsheet Link 的应用。

【例 11-18】 一个 Spreadsheet Link 的实例。

本例用数据表执行方式运行，使用 Excel 工作表组织和显示数据，该实例 ExliSample.xls 位于 MATLAB 安装路径下的\toolbox\exlink 子目录中。

启动 Excel、Spreadsheet Link 和 MATLAB，打开示例文件 ExliSample.xls。

单击 ExliSample.xls 中的 Sheet1 标签，可以看到数据表中包含一个被命名为 DATA 的数据区 A4:C28，如图 11-17 所示。

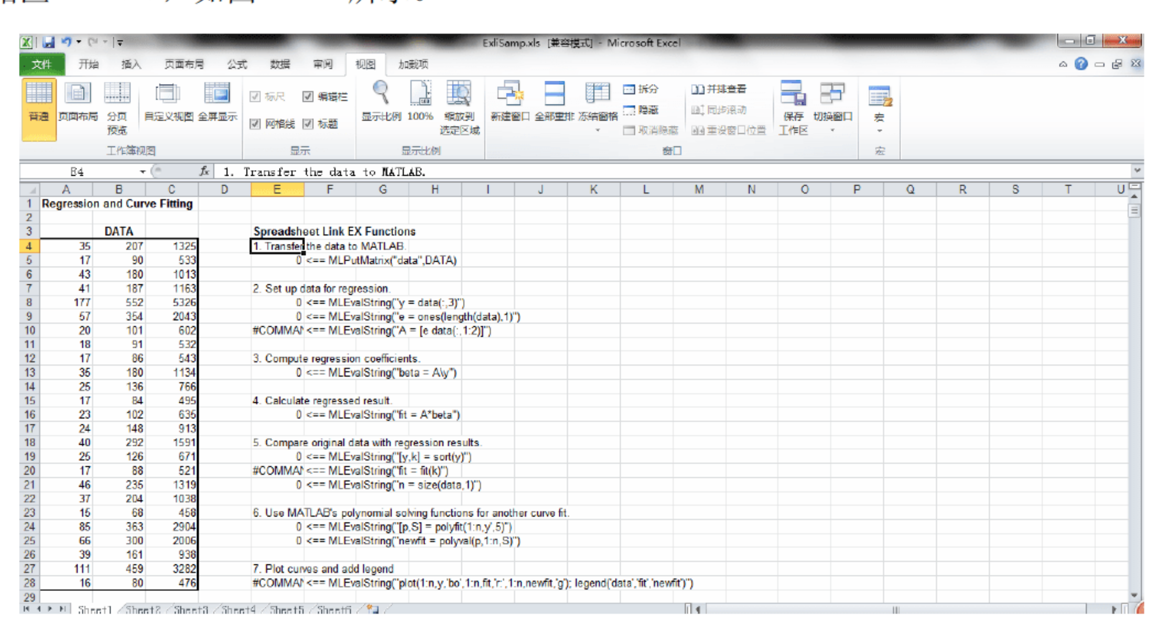


图 11-17　ExliSample.xls 中的 Sheet1 数据页

该数据区包含了本例的数据集，按照下面的步骤进行操作。

（1）选中 Sheet1 页的 E5 单元格，按“F2”键，然后按回车键执行 Excel Link 函数 MLPutMatrix(“data”, DATA)，将 DATA 复制到 MATLAB 的 data 变量中，data 包含 3 个变量的 25 次观测值。

（2）选中 E28 单元格，按“F2”键，然后按回车键，对 E9 和 E10 单元格重复上面的操作。这些 Excel Link 函数让 MATLAB 以第 1 列和第 2 列的数据对应的变量为自变量，以第 3 列数据对应的变量为应变量进行回归。

（3）运行 E13 中的函数。使用 MATLAB 的“\”操作符计算回归系数，以便求解线性方程 A*beta=y。

（4）运行 E16 中的函数。MATLAB 使用矩阵-向量乘法生成回归结果（fit）。

（5）运行 E19、E20、E21 中的函数。这些函数对原始数据与 fit 进行比较，将数据按升序排列，对 fit 采用相同的 permutationg，并创建一个表示观测数据的标量。

（6）运行 E24、E25 中的函数。用于拟合多项式，plotfit 函数生成一个 5 阶多项式 polyval，其次计算每个数据点上多项式的结果，确定拟合精度（newfit）;

（7）运行 E28 中的函数。添加图列得到用 MATLAB 的 plot 函数画出的图形，其中“○”代表原始数据，“……”代表回归结果，“——”代表多项式结果，可得到如图 11-18 所示的结果。

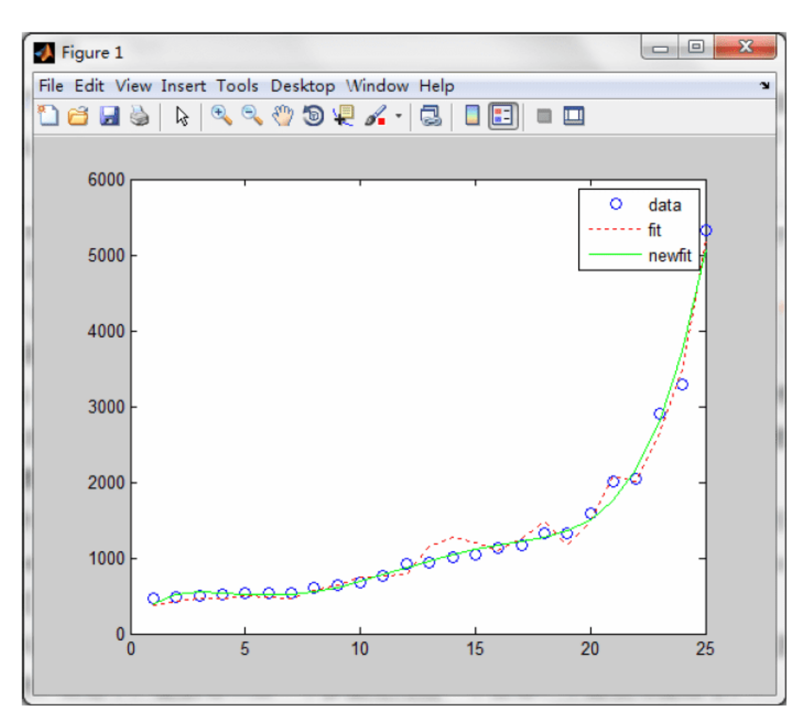


图 11-18　回归和曲线拟合结果显示

对图 11-18 中的 data、fit 和 newfit 三条曲线进行比较，可以发现数据具有很强的相关性，不是线性独立的，拟合曲线和原始数据并不是十分吻合，而 5 阶多项式拟合则显示了更加精确的数学模型。

还需要说明以下几点。

- Excel 本身也具有强大的数据显示功能，在 MATLAB 和 Excel 混合使用时可以充分利用它的这一优势。
- Spreadsheet Link 函数名对字母的大小写不作区分，如 MLPutMatrix 与 mlptmatrix 等价，而 MATLAB 函数名是区分大小写的。
- Excel 表单的执行语句一般加等号，如“=MLGetMatrix(" y" ," C1:L10")”，并且函数的参数用圆括号括起来。在宏中，函数名和第一个参数之间隔一个空格，不能使用圆括号。
- 单击执行语句的单元显示对应的语句，执行完毕后单元显示值为 0。
- 执行语句中的标点符号和 MATLAB 中的相同，都必须在英文状态下输入。
- 在 Excel 的电子表格中直接输入函数即可，不要使用 Excel 的函数向导，否则会产生不可预料的结果。
- Excel Link 只处理 MATLAB 二维的数值数组和一维的字符数组（字符串）及二维的细胞数组，而不能操作 MATLAB 的多维数组和结构体。

11.4 编译器

希望 MATLAB 能够更快地运行程序代码，或者希望获得可摆脱 MATLAB 运行环境而独立运行的可执行文件。为满足用户这方面的需要，出现了 MATLAB 编译器。

MATLAB 编译器可以编译 M 文件、MEX 文件及其他的 MATLAB 代码，支持 MATLAB 所有的特性，包括对象、私有函数和方法。

编译器可以产生以下几种应用程序。

独立运行的程序，可以在没有安装 MATLAB 7.1 的机器上运行。

C 和 C++共享库（在 Microsoft Windows 操作系统中为动态链接库 DLL）：这些共享库可以在没有安装 MATLAB 2010 的用户机器上运行。

前面介绍的 MATLAB 程序都是运行在 MATLAB 平台上的，本节将介绍如何得到由 MATLAB 平台开发，经编译器编译后可以独立于 MATLAB 平台运行的程序。

11.4.1 编译器的安装和配置

在使用编译器前，需要进行安装和配置，在命令窗口输入如下语句。

```
mbuild-setup
```

命令窗口显示如下提示。

```
Please choose your compiler for building standalone MATLAB applications:
Would you like mbuild to locate installed compilers [y]/n?
```

选择 y 之后，命令窗口的输出结果如下。

```
Select a compiler:
[1] Lcc-win32 C 2.4.1 in D:\MATLAB\R2010a\sys\lcc
[0] None
Compiler:
```

其含义是“对 mbuild 已安装的编译器进行选择后，运行程序”。

这里列出了 MATLAB 支持的通用编译器。选择 0 后，命令窗口中的输出结果如下所示。

```
mbuild: No compiler selected. No action taken.
```

再次在命令窗口输入如下语句。

```
mbuild-setup
```

命令窗口中的输出结果如下。

```
Select a compiler:
[1] Lcc-win32 C 2.4.1 in D:\MATLAB\R2010a\sys\lcc
```

```
[0] None
Compiler:
```

此时显示系统中已经安装的编译器，并且列出安装的目录。Lcc-win32 C2.4.1 是 MATLAB 自带的 C 编译器，不能用来编译 C++。选择 1 后，命令窗口中的输出结果如下所示。

```
Are these correct [y]/n?
```

选择 y 后，输出结果显示如下所示。

```
Trying to update options file:
C:\Users\admin\AppData\Roaming\MathWorks\MATLAB\R2010a\compopts.bat
From template:     D:\MATLAB\R2010a\bin\win32\mbuildopts\lcccompp.bat
Done . . .
```

至此，完成了编译器的安装和配置。

另外，为了能够使用 MATLAB 编译器生成的组件，必须安装 MCR。可以先将 MATLAB 安装路径下的\toolbox\compiler\deploy\win32 子目录中的文件 MCRInstaller.exe 复制到另一路径，然后双击进行安装，直到提示安装结束。

11.4.2 编译命令

在 MATLAB 中使用 mcc 命令对 MATLAB 各类代码进行编译，其具体用法如下所示。

```
MCC[-options]fun[fun2…]
```

其中，options 为选项，fun 和 fun2 为 MATLAB 代码文件，最常用的几种形式如下。

- **Ncc-m myfun** 将 M 文件生成独立运行的同名 exe 文件。
- **Ncc-m myfun1 myfun2** 将 M 文件主函数生成可独立运行的同名 exe 文件。
- **Ncc-W lib:liba-T link:lib a0 a1** 将两个 M 文件生成名为 liba 的 C 共享库。
- **Ncc-W cpplib:liba-T link:lib a0 a1** 将两个 M 文件生成名为 liba 的 C++共享库。

下面通过几个实例具体介绍编译命令的使用方法。

【例 11-19】 编译函数求矩阵的特征值，函数文件 myeig.m 中具体的函数指令如下所示。

```
function eigA=myeig(u1,u2,u3,u4)
A=[eval(u1)eval(u2);eval(u3)eval(u4)]    %数据类型强制转换
eigA=eig(A)
```

在命令窗口输入如下语句。

```
mcc -m myeig
```

执行后，myeig.m 所在目录下增加了 myeig.exe、myeig.prj、myeig_main.c、myeig_delay_load.c 和 myeig_mcc_component_data.c 文件。

在命令窗口输入如下语句。

```
type myeig_main.c
```

命令窗口中的输出结果如下。

```
/*
 * MATLAB Compiler: 4.13 (R2010a)
 * Date: Thu Oct 03 09:30:03 2013
 * Arguments: "-B" "macro_default" "-m" "-W" "main" "-T" "link:exe" "myeig"
 */

#include <stdio.h>
#include "mclmcrrt.h"
#ifdef   cplusplus
extern "C" {
#endif

extern mclComponentData   MCC myeig component data;

#ifdef   cplusplus
}
#endif

static HMCRINSTANCE _mcr_inst = NULL;

#ifdef __cplusplus
extern "C" {
#endif

static int mclDefaultPrintHandler(const char *s)
{
  return mclWrite(1 /* stdout */, s, sizeof(char)*strlen(s));
}

#ifdef __cplusplus
} /* End extern "C" block */
#endif

#ifdef __cplusplus
extern "C" {
#endif

static int mclDefaultErrorHandler(const char *s)
{
  int written = 0;
  size_t len = 0;
  len = strlen(s);
  written = mclWrite(2 /* stderr */, s, sizeof(char)*len);
  if (len > 0 && s[ len-1 ] != '\n')
    written += mclWrite(2 /* stderr */, "\n", sizeof(char));
  return written;
}
```

```
#ifdef   cplusplus
} /* End extern "C" block */
#endif

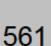
/* This symbol is defined in shared libraries. Define it here
 * (to nothing) in case this isn't a shared library.
 */
#ifndef LIB myeig C API
#define LIB_myeig_C_API /* No special import/export declaration */
#endif

LIB myeig C API
bool MW CALL CONV myeigInitializeWithHandlers(
    mclOutputHandlerFcn error handler,
    mclOutputHandlerFcn print handler
)
{
  if ( mcr inst != NULL)
    return true;
  if (!mclmcrInitialize())
    return false;
  if (!mclInitializeComponentInstanceWithEmbeddedCTF(& mcr inst, &  MCC
  myeig component data,
   true, NoObjectType, ExeTarget, error handler, print handler, 44744, NULL))
    return false;
  return true;
}

LIB myeig C API
bool MW CALL CONV myeigInitialize(void)
{
  return myeigInitializeWithHandlers(mclDefaultErrorHandler,
                                     mclDefaultPrintHandler);
}

LIB myeig C API
void MW CALL CONV myeigTerminate(void)
{
  if ( mcr inst != NULL)
    mclTerminateInstance(& mcr inst);
}

int run_main(int argc, const char **argv)
{
  int _retval;
  /* Generate and populate the path_to_component. */
  char path_to_component[(PATH_MAX*2)+1];
  separatePathName(argv[0], path_to_component, (PATH_MAX*2)+1);
  __MCC_myeig_component_data.path_to_component = path_to_component;
  if (!myeigInitialize()) {
```

```
      return -1;
    }
    argc = mclSetCmdLineUserData(mclGetID( mcr inst), argc, argv);
     retval = mclMain( mcr inst, argc, argv, "myeig", 1);
    if (_retval == 0 /* no error */) mclWaitForFiguresToDie(NULL);
    myeigTerminate();
    mclTerminateApplication();
    return  retval;
  }

  int main(int argc, const char **argv)
  {
    if (!mclInitializeApplication(  MCC myeig component data.runtime options,
         MCC myeig component data.runtime option count))
      return 0;

    return mclRunMain(run main, argc, argv);
  }
```

由上可见，编译器的版本为 V4.13，然后进入命令提示符模式（DOS 窗口），并将 myeig.m 所在目录设置为当前目录，在 DOS 窗口中输入如下语句。

```
  myeig 1.3  2.5  3.9  4.0
```

DOS 窗口中的运行结果如图 11-19 所示。

图 11-19　在 DOS 窗口得到的结果

这里需要说明的是，语句“A=[eval(u1)eval(u2);eval(u3),eval(u4)]中的强制转换对于编译来说非常重要，否则系统会将输入按照字符类型进行处理。

11.4.3　项目开发工具

下面通过一个实例介绍项目开发工具的应用。

【例 11-20】 利用项目开发工具构建 C 共享库，具体步骤和操作如下。

编写 myprjf.m 程序代码如下。

```
function y=myprjf(x)
y=sin(eval(x))
```

在 MATLAB 主窗口中选择【File】/【New】/【Deployment Project】菜单命令，打开如图 11-20 所示的界面。

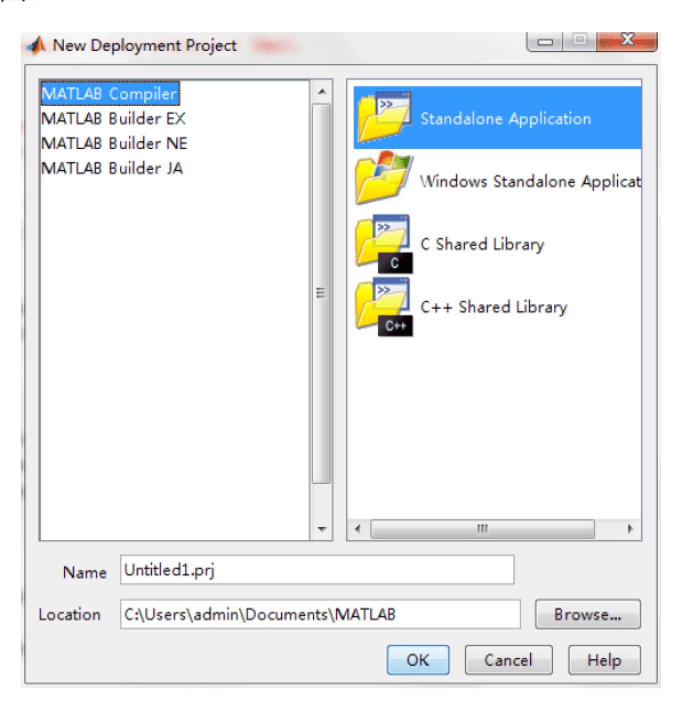


图 11-20　项目开发工具开始界面

在“Name”框中填写项目名称（这里为 mypro.prj），在“location”框中选择保存的路径，然后单击“OK”按钮，出现如图 11-21 所示的界面。

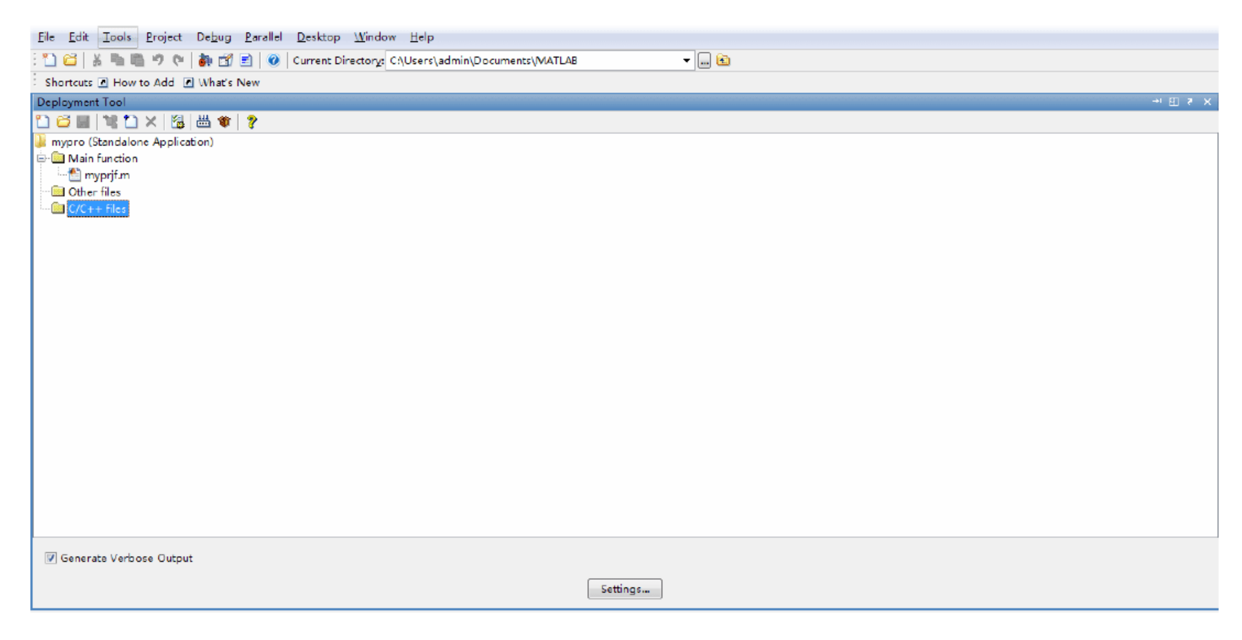


图 11-21　项目开发工具界面

右击“Main function”，打开添加文件对话框并选择 myprjf.m 文件，并单击图标，可以打开如图 11-22 所示的界面。

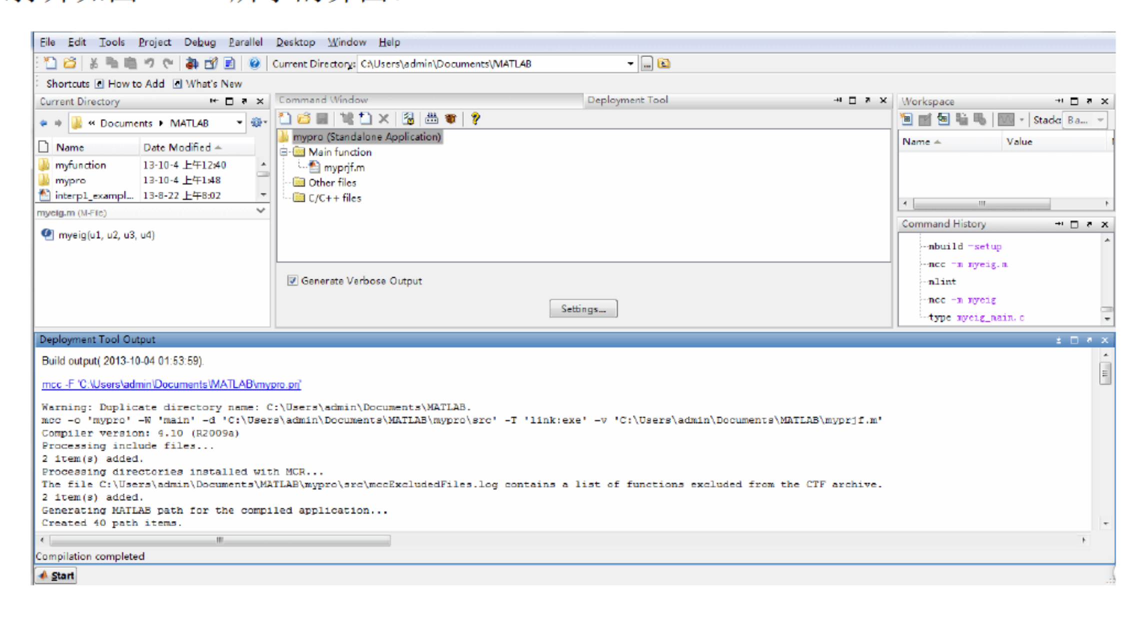


图 11-22　编译后的结果

编译后在指定目录下增加了 mypro.prj 文件和子目录 mypro，目录 mypro 下还包含两个下一级目录 distrib 和 src。

最后进入 DOS 窗口，并将当前目录设置为 myprjf.m 所在目录下的\mypro\distrib 目录（或\mypro\src 目录），再如图 11-23 所示输入如下语句。

```
mypro pi
```

在 DOS 窗口中的运行结果如图 11-24 所示。

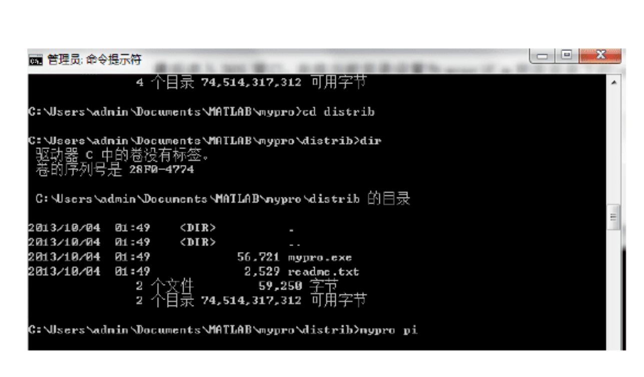


图 11-23　在 DOS 窗口输入命令

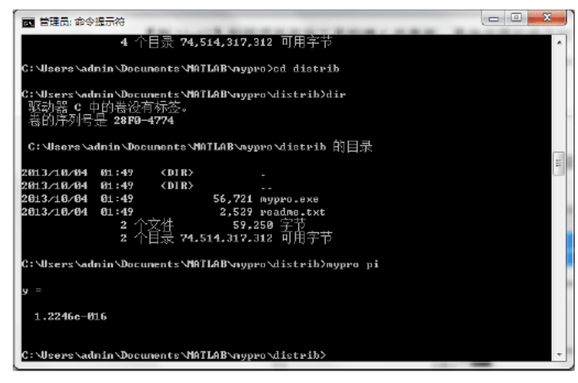


图 11-24　在 DOS 窗口得到的结果

11.5　MATLAB 与 C/C++语言混合使用

MATLAB 具有很强的数值计算和分析等能力，而 C/C++是目前最为流行的高级程序设计语言，为使两者能结合运用，实现优势互补，将获得极大的效益。MathWorks 公司提供

了 MATLAB 和 C、C++的接口。通过接口，用户既可在 C 程序中调用 MATLAB 的函数，也可在 MATLAB 中调用 C 或 C++程序，从而实现 MATLAB 和 C、C++的混合编程，两者互补结合的混合编程在科学研究和工程实践中具有非常重要的意义。本节将介绍 MATLAB 与 C/C++语言混合使用是相关知识。

11.5.1　MATLAB C/C++编译器的设置（MEX）

MATLAB 与 C 语言的混合使用包括两个方面，即在 MATLAB 平台上调用 C 语言资源，以及在 C 语言平台上调用 MATLAB 资源。

在 MATLAB 中使用 mex 命令将 C 语言文件编译成 MEX 文件形式的共享库，以便 MATLAB 调用。尽管在 MATLAB 中调用 MEX 文件会比较简单，但是一般格式编写的 C 语言程序代码并不能直接编译成可以被 MATLAB 调用的 MEX 文件，只有符合某种特殊格式的 C 程序代码才能编译成为 MEX 文件。

利用 MATLAB 编译器可以将 C/C++程序编译为 MEX 文件供 MATLAB 直接调用。此外，MATLAB 也可以将*.m 函数文件直接编译为独立可执行文件、动态链接库和 C/C++语言程序。MATLAB C/C++编译器的设置包括 mbuild –setup 和 mex –setup 两个步骤。

在 MATLAB 命令窗口输入 mex –setup，并根据提示选择合适的选项，具体如下。

```
>> mex -setup
Please choose your compiler for building external interface (MEX) files:
Would you like mex to locate installed compilers [y]/n? y
Select a compiler:
[1] Lcc-win32 C 2.4.1 in D:\MATLAB\R2010b\sys\lcc
[0] None

Compiler: 1
Please verify your choices:
Compiler: Lcc-win32 C 2.4.1
Location: D:\MATLAB\R2010b\sys\lcc
Are these correct [y]/n? y
Trying to update options file: C:\Users\admin\AppData\Roaming\MathWorks\
MATLAB\R2010b\mexopts.bat
From template:          D:\MATLAB\R2010b\bin\win32\mexopts\lccopts.bat
Done . . .
**************************************************************************
Warning: The MATLAB C and Fortran API has changed to support MATLAB
variables with more than 2^32-1 elements.  In the near future
you will be required to update your code to utilize the new
API. You can find more information about this at:
         http://www.mathworks.com/support/solutions/data/1-5C27B9.html?
         solution=1-5C27B9
Building with the -largeArrayDims option enables the new API.
**************************************************************************
```

11.5.2　MATLAB 中调用 C/C++程序-MEX 文件

1. MEX 文件的用途

在 MATLAB 开发环境中调用 C/C++等外部程序需要借助编译器将 C/C++代码编译为

MEX 文件以后才能实现，其中，MEX 文件包含有 MATLAB 解释器可以动态装载和执行的动态链接模块。在 Windows 平台下，MEX 文件是以动态链接库（根据 MATLAB 版本的不同，扩展名为*.dll、*.mexw32、*.mexw64）的形式存在。通过用 C/C++语言编写代码，然后通过 MATLAB 编译器将其编译为 MEX 文件可以完成如下功能。

1）加快程序的执行速度

采用 MATLAB 开发数值计算程序时，如果其中有的模块在 MATLAB 中的执行效率很低（如循环），可以采用 MEX 文件的方式用 C/C++语言来实现这些执行效率比较低的模块，从而提高整个程序的执行速度。

2）利用 MATLAB 作为 C/C++语言开发的调试环境

在 MATLAB 中进行数据显示是非常方便的，C/C++语言编写的数值计算程序在其他开发环境中调试时数据显示不方便，因此，可以通过 MEX 文件这种方式在 MATLAB 环境下进行调试。尤其是有大量数据需要处理的情况下，用 MATLAB 观察其中间结果是非常方便的。

3）扩展 MATLAB 的功能

MATLAB 具有强大的矩阵运算能力并且拥有丰富的工具箱，但是在有些方面的功能还显得比较薄弱，如硬件设备接口操作、图形化程序设计等方面，用户可以通过 MEX 文件利用 C/C++语言扩展 MATLAB 的薄弱环节，以满足设计的需求。

2. MEX 文件与 M 文件的关系

只要按照 MATLAB 规定的书写格式编写 C/C++程序，那么生成的 MEX 文件的调用方法和一般的 MATLAB 函数完全相同，在使用 MEX 文件的时候应该注意以下两点。

1）MEX 文件与 M 文件的优先级

MEX 文件的优先级比一般的 MATLAB 函数文件高，因而如果一个目录下存在名称相同的 MEX 文件和 MATLAB 函数文件，则 MEX 文件会被优先调用，而相应的 MATLAB 函数文件由于执行优先级较低不会被调用。

2）通过 M 文件存储 MEX 文件帮助信息

如果想让 MEX 文件像一般的 MATLAB 函数那样，则用 help 命令可以看到这个函数的帮助信息，可以在 MEX 文件相同目录下放置名称相同的 M 文件，并将帮助信息放在 M 文件中的方法来解决，因为 help 命令只能显示 M 文件的帮助信息。

3. 一个简单的 MEX 文件实例

下面举一个简单的例子说明 MEX 文件的编写和创建过程，具体步骤如下。

1）编写 C 语言的文件

```
/*helloworld.c*/
    #include "mex.h"
    void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray *prhs[])
    {
      mexPrintf("Hello World!\n");
}
```

2）将 C 语言文件编译为 MEX 文件

在 MATLAB 命令行下，执行命令 mex <filename>.c 将编写的 C 语言文件编译为 MEX

文件。如果 C 语言文件名为 helloworld.c，用 mex helloworld.c 将其编译为 MEX 文件。编译成功以后，可以在 helloworld.c 文件相同的目录下找到 helloworld.wmex32（或 helloworld.mexw64）文件，这就是编译成功的 MEX 文件。

3）执行 MEX 文件

在 MATLAB 命令行中输入 helloworld，即可执行编译好的 helloworld MEX 文件，如下所示。

```
>>helloworld
Hello World!
>>
```

4）创建 MEX 文件的帮助文件

如果想为 MEX 文件创建帮助文件，可以创建与 MEX 文件同名的.M 文件。如果为 helloworld MEX 文件创建帮助文件，可以创建名称为 helloworld.m 的 M 文件，其代码如下。

```
%保存为 helloworld.m 文件
%function [] = helloworld()
%MEX 文件第一个实例
```

然后在命令行窗口输入 help 命令可以查看创建完成的帮助文件，如下所示。

```
>>help helloworld
function [] = helloworld()
MEX 文件第一个实例
```

4. MEX 文件结构说明

用 C/C++语言编写的 MEX 文件源代码时，必须创建 mexFunction 函数。mexFunction 函数的作用与 C/C++语言程序设计中的 main 函数的功能类似。如果 main 函数提供的是操作系统与 C 语言子程序之间的接口，mexFunction 函数的作用则是 MATLAB 与 C/C++语言子程序之间的接口。另外，用 C/C++语言编写 MEX 文件源代码时，需要包含"mex.h"头文件，如下所示。

```
#include "mex.h"
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]);
```

其中，nlhs 为输出参数的个数，plhs 为输出参数的 mxArray 数组，nrhs 为输入参数的个数，prhs 为输入参数的 mxArray 数组。

在 MATLAB 中，所有的数据类型都使用 mxArray 结构来表示。通过接口函数 mexFunction，可以和 MATLAB 环境进行数据交换。MATLAB 与 mexFunction 数据交互的过程如图 11-25 所示。从该图可以看出，输入参数用 nrhs 和 prhs 这两个量来描述。prhs 是一个 mxArray 的指针数组，而 nrhs 表示这个数组的大小，即输入参数的个数。同样，输出参数用 plhs 和 nlhs 描述，其中，plhs 是一个 mxArray 指针数组，而 nlhs 则表示 plhs 的大小，即输出参数的个数。MEX 文件的一般结构如图 11-25 所示。假设编写的 MEX 函数（myfunc）输入参数为 3 个，输出参数为 2 个，在 MATLAB 命令行中执行 MEX 文件的格式为

```
>> [Out1,Out2] = myfunc(In1,In2,In3);
```

而在 MEX 文件中，In1 对应的 mxArray 为 prhs[0]，In2 对应的 mxArray 为 prhs[1]，In3 对应的 mxArray 为 prhs[2]。在 mexFunction 中，用户通过 nrhs 判断输入参数的个数；通过 nlhs 判断输出参数的个数，然后需要创建 plhs[0]和 plhs[1]两个 mxArray 作为输出数据。在 MEX 文件中一个 mxArray 变量和 MATLAB 环境中的一个阵列变量等同。

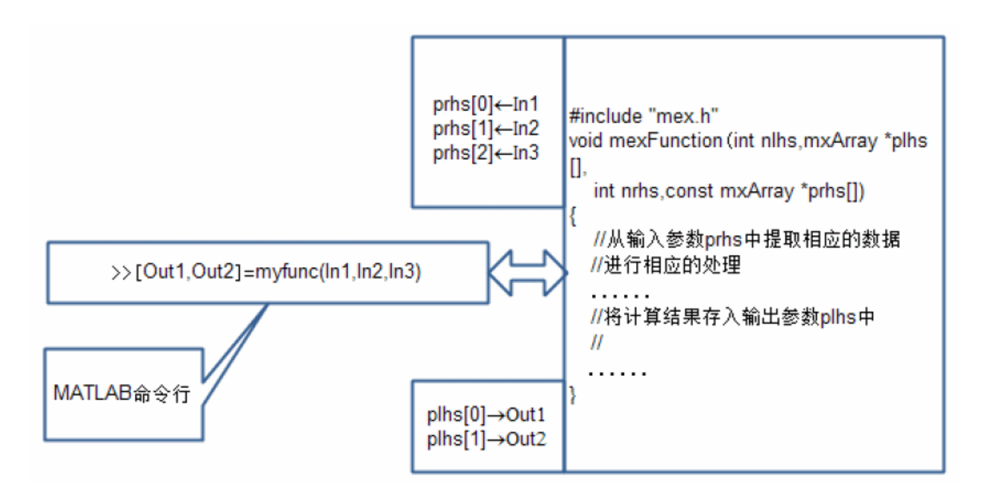


图 11-25　mexFunction 与 MATLAB 的数据交换

5．编译 MEX 文件

在 MATLAB 命令行中通过 mex 命令编译 MEX 文件，假设要编译 filename.c 文件，只需要在命令窗口中输入 mex filename.c 命令即可。mex 命令有很多选项，如果用户有特殊的应用，则可以在调用 mex 命令时设置这些选项，如采用-largeArrayDims 选项意味着 MEX 文件中可以使用长度大于 2^31-1 的阵列，其他更多的 mex 命令选项用户可以查看 MATLAB 帮助。

6．采用 C++创建 MEX 文件

采用 C++构建 MEX 文件的方法与 C 语言类似，可以通过创建 C++类完成数值计算、图形界面显示，外部数据接口操作等功能。注意，编译 C++构建的 MEX 文件时可能需要附加多个源文件。为了说明此文件，编写了一个 C++版的 helloworld MEX 文件如下。其中，hw.cpp 为 MEX 文件的主程序，HelloWorld.h 和 HelloWorld.cpp 为自定义的 HelloWorld 类。在 MATLAB 命令行窗口中输入如下命令完成 hw MEX 文件的编译。

```
mex hw.cpp HelloWorld.cpp
```

hw.cpp 如下所示。

```
//保存为 hw.cpp
#include "mex.h"
#include "HelloWorld.h"
void mexFunction(int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[])
{
  HelloWorld hw;
  mexPrint("Hello World!\n");
}
//保存为 HelloWorld.h
#ifndef _HELLO_WORLD
```

```
#define  HELLO WORLD
#inclue "mex.h"
class HelloWorld
{
  public:
    HelloWorld(void);
    ~HelloWorld(void);
  public:
    void Print();
};

//保存为 HelloWorld.cpp
#include "HelloWorld.h"
HelloWorld::HelloWorld(void)
{
}

void HelloWorld::Print()
{
  mexPrintf("Hello World!\n");
}

HelloWorld::~HelloWorld(void)
{
}
```

11.5.3 MATLAB 与 C 语言混合编程常用的数据类型

1. size_t 类型

不同的机器和编译器的 size_t 的定义是不同的，如下面这几种定义都是可能出现的。

```
typedef unsigned int size t;
typedef int size t;
typedef short size t;
typedef unsigned short size_t;
```

当然也可能出现其他定义。总之，size_t 主要用来作为 sizeof 函数的返回类型，或者作为描述变量及内存长度的一种类型。之所以采用 size_t 而不是直接采用 int、long、short 或其他整型，是为了程序代码的通用性，假如程序执行的平台变了，只要改变 size_t 的定义然后重新编译代码即可。

2. MATLAB C 语言接口中的特殊类型

几个特殊的数据类型包括：

```
mwIndex, mwSize, mxChar, mxLogical, mxClassID, mxComplexity
```

下面分别对以上类型进行介绍。

1）mwIndex 和 mwSize

mwIndex 用以表示阵列索引；mwSize 用以表示大小，比如，阵列维数或阵列长度。

mwIndex 和 mwSize 的定义在 extern/include/tmwtypes.h 文件中，定义如下。

```
#indef MX COMPAT 32
typedef int mwSize;
typedef int mwIndex;
typedef int mwSignedIndex;
#else
typedef size t mwSize;
typedef size t mwIndex;
typedef ptrdiff t mwSignedIndex;
#endif
```

从这个定义可以看出，当 MX_COMPAT_32 宏定义存在时，MATLAB 对 mwIndex 和 mwSize 变量的定义实际上是 int 类型；当 MX_COMPAT_32 宏定义不存在时，MATLAB 对 mwIndex 和 mwSize 变量的定义实际上是 size_t 类型。

MATLAB 之所以推出这两个变量而不再沿用原来的 int 类型表示阵列索引和阵列长度，这是为了适应 64 位处理器和处理更大规模阵列的需要。编译 MATLAB MEX 文件的 MEX 命令增加了-largeArrayDims 选项，如果编译 MEX 文件此时不带此选项，则此时 mwSize 和 mwIndex 定义为 int；如果编译 MEX 文件时带有此选项，则此时 mwSize 和 mwIndex 定义为 size_t。从 MATLAB 2006b 开始，MATLAB 对稀疏矩阵的存储方式与以前版本发生了变化，只有使用-largeArrayDims 选项（在 MATLAB 2010 之后的版本中，MathWorks 公司计划将-largeArrayDims 作为编译 MEX 文件的默认选项）。

2）mxChar 和 mxLogical

mxChar 和 mxLogical 是 MATLAB 自定义的两个数据类型，在 extern\include/matrix.h 中，可以找到它们的定义，即

```
typedef bool mxLogical;
typedef char16_t mxChar;
```

可以看出，mxLogical 实际上就是 bool 类型；而 mxChar 类型实际上就是 16 位字符类型（MATLAB 中所有的字符都采用 16 位双字节存储）。

3）mxClassID

mxClassID 用来描述 MATLAB 的阵列类型，所有的 MATLAB 阵列类型都有一个对应的枚举量 ID，比如元组阵列的 ID 为 mxCELL_CLASS。因而，使用 mxClassID 类型可以很方便地判断某一 MATLAB 阵列（mxArray 表示）属于哪一种类型。在文件 extern\include\matrix.h 中，对 mxClassID 类型的定义如下。

```
typedef enum
{
  mxUNKNOWN CLASS = 0,
  mxCELL CLASS,
  mxSTRUCT CLASS,
  mxLOGICAL CLASS,
  mxCHAR CLASS,
  mxVOID CLASS,
  mxDOUBLE_CLASS,
```

```
  mxSINGLE CLASS,
  mxINT8 CLASS,
  mxUINT8 CLASS,
  mxINT16 CLASS,
  mxUINT16 CLASS,
  mxINT32 CLASS,
  mxUINT32 CLASS,
  mxINT64 CLASS,
  mxUINT64 CLASS,
  mxFUNCTION CLASS,
  mxOPAQUE CLASS,
  mxOBJECT CLASS,
#if defined( LP64) || defined( WIN64)
  mxINDEX CLASS = mxUINT64 CLASS,
#else
  mxINDEX CLASS = mxUINT32 CLASS,
#endif
/*已经废止，用户不可再使用此类型*/
  mxSPARSE_CLASS = mxVOID_CLASS
}
```

4）mxComplexity

mxComplexity 类型用以区分浮点数值阵列是实数还是复数，它也是一个枚举类型，在文件 extern\include\matrix.h 中，对其定义如下所示。

```
typedef enum{
       mxREAL,
       mxCOMPLEX
} mxComplexity;
```

假设用户需要创建一个 5 行 3 列的复数双精度数值阵列，用户需要传递 mxCOMPLEX 给 mxCreateDoubleMatrix 函数，如下所示。

```
mxArray *pa;
pa = mxCreateDoubleMatrix(5,3,mxCOMPLEX);
```

5）mxArray

从图 11-25 中可以看出，用 C\C++语言编写 MEX 文件的一个关键之处就在于 mexFunction 函数中关于 MATLAB 与 C\C++代码模块的数据交互问题。MATLAB 所有的数据类型都可以用 mxArray 描述，并且 mexFunction 函数的所有输入/输出参数都是采用 mxArray 的形式来实现的。那么如何将 mxArray 转换为 C\C++语言可以直接使用的基本数据类型（如 double，int 等），或者如何将 C\C++语言的基本数据类型转换为 mxArray 类型？在 matrix.h 文件中可以找到 mxArray 的定义如下。

```
typedef struct mxArray_tag mxArray;
```

由此可见，mxArray 实际上是一个结构体，由于 MATLAB 已经隐藏了 mxArray_tag 的定义，所以无从得知 mxArray 的具体定义。但通过一个 C 语言结构体表示诸如数值阵列、字符阵列、元组阵列和结构体阵列等多种类型的 MATLAB 数据并非易事，这个问题

MATLAB 通过提供一系列操作 mxArray 的 API 函数得以解决。MATLAB 与 C\C++混合编程时经常需要调用 MATLAB 提供的 API 函数，其中，以 mx 开头的 MATLAB API 函数主要是提供对 mxArray 进行操作的函数，而以 mex 开头的 MATLAB API 函数只能在 MEX 文件中应用，以 mx 开头的 MATLAB API 函数可以在其他应用，如从 C\C++调用 MATLAB 程序中应用。所以 mx 函数在 MATLAB 与 C\C++混合程序设计中具有重要应用。

11.5.4 操作 MATLAB 阵列 mxArray 的 mx 函数

1. 数值阵列操作函数

1）创建数值阵列

数值阵列创建函数分为三类：创建双精度型的数值阵列，创建任意类型的数值阵列，创建数值阵列标量（即数值阵列）。由于双精度类型数值阵列和数值阵列标量在应用中最常用，所以 MATLAB 提供了 mxCreateDoubleMatrix 和 mxCreateDoubleScalar 来创建双精度类型数值阵列和数值阵列标量。如果要创建其他类型数值阵列或创建多维数值阵列，则需要采用 MATLAB 提供的 mxCreateNumericArray 和 mxCreateNumericMatrix 函数。数值阵列创建函数的详细说明如下。

```
mxArray *mxCreateDoubleMatrix(mwSize m, mwSize n, mxComplexity ComplexFlag);
```

函数功能：创建双精度矩阵()。

参数说明：m 表示矩阵的行数；n 表示矩阵的列数；ComplexFlag 表示数值阵列的类型，mxComplexity 是一个枚举类型，其中 mxREAL 表示当前 mxArray 是一个实数数组，没有虚部。mxCOMPLEX 表示当前 mxArray 是一个复数数组，存在虚部，即 mxGetPi 函数返回不为空。

返回值：返回创建的二维双精度数值阵列的 mxArray 指针。

```
mxArray *mxCreateDoubleScalar(double value)
```

函数功能：创建双精度标量（数值阵列）。mxCreateDoubleScalar 函数的功能与下面的语句相同，只不过书写起来比较简洁。

参数说明：value 表示生成的双精度标量的值。

返回值：返回生成的双精度数值阵列。

```
mxArray *mxCreateNumericArray(mwSize ndim, const mwSize *dims,mxClassID
class, mxComplexity ComplexFlag);
```

函数功能：创建任意维数任意类型的数值阵列。

参数说明：ndim 表示待创建数值阵列的维数；*dims 表示待创建数值阵列各维大小，其中 dims 是一维数组，数组长度为 ndim；class 表示创建的数值阵列的类型；mxClassID 是一个枚举类型，用以表示 MATLAB 开发环境中当前支持的所有 MATLAB 阵列类型。通过函数 mxGetClassID 可以得到 mxArray 的数组类型；ComplexFlag 表示数值阵列是实数型还是复数型。

返回值：创建的 MATLAB 数值阵列 mxArray 指针。

```
mxArray *mxCreateNumericMatrix(mwSize m, mwSize n, mxClassID class,
mxComplexity ComplexFlag);
```

函数功能：创建任意类型数值矩阵（二维数值阵列）。

参数说明：m 表示数值矩阵的行数；n 表示数值矩阵的列数；class 表示数值矩阵的类型；ComplexFlag 表示数值矩阵是实数型还是复数型。

返回值：创建的 MATLAB 数值矩阵的 mxArray 指针。

2）数值阵列数据操作

无论什么类型的数值阵列，在 MEX 文件均通过 mxArray 表示，但是必须获取 mxArray 对应的数值阵列的数据才能完成数据处理工作。对于数值阵列的数据操作而言，主要关心两个方面：一是数据类型，包括双精度、单精度、整型等；二是数值阵列式实数型还是复数型，mxArray 对应的实数型数据和复数型数据分别独立保存。对于双精度（double）型数值阵列，MATLAB 提供了 mxGetPr、mxGetPi、mxSetPr 和 mxSetPi 四个函数进行操作。此外，用户可以通过 mxGetData、mxGetImagData、mxSetData、mxSetImagData 四个函数对任意数据进行操作。数值阵列数据操作函数说明如下。

```
double *mxGetPr(const mxArray *array_ptr);
```

函数功能：得到 MATLAB 双精度型数值阵列的实部数据指针。

```
double *mxGetPi(const mxArray *array_ptr);
```

函数功能：得到 MATLAB 双精度型数值阵列的虚部数据指针。

```
void mxSetPr(mxArray *array_ptr, double *pr);
```

函数功能：设置双精度型数值阵列的实部数据指针。

参数说明：array_ptr 表示 MATLAB 双精度型数值阵列的 mxArray 指针，pr 表示指向实部数据的指针。

```
void mxSetPi(mxArray *array_ptr, double *pi);
```

函数功能：设置双精度型数值阵列的虚部数据。

参数说明：array_ptr 表示 MATLAB 双精度型数值阵列的 mxArray 指针，pi 表示指向虚部数据的指针。

```
void *mxGetData(const mxArray *array_ptr);
```

函数功能：得到非双精度型数值阵列实部数据的指针。

参数说明：array_ptr 表示 MATLAB 数值阵列的 mxArray 指针。

```
void *mxGetImagData(const mxArray *array_ptr);
```

函数功能：得到非双精度型数值阵列虚部数据的指针。

参数说明：array_ptr 表示 MATLAB 数值阵列的 mxArray 指针。

```
void mxSetData(mxArray *array_ptr, void *data_ptr);
```

函数功能：设置 MATLAB 非双精度型数值阵列的实部数据。

参数说明：array_ptr 表示 MATLAB 非双精度型数值阵列，data_ptr 表示指向实部数据的指针。

附加说明：mxSetData 与 mxSetPr 的函数功能相似，用于设置非双精度型数值阵列的实部数据。

```
void mxSetImagData(mxArray *array_ptr, void *pi);
```

函数功能：设置非双精度型数值阵列的虚部数据。

参数说明：array_ptr 表示 MATLAB 非双精度型数值阵列的 mxArray 指针，pi 表示指向虚部数据的指针。

附加说明：mxSetImagData 与 mxSetPi 的函数功能相似，用于设置非双精度型数值阵列的虚部数据。

3）标量数据操作

```
double mxGetScalar(const mxArray *array_ptr);
```

函数功能：得到 MATLAB 数值阵列第一个数据元素的值。

返回值：返回 array_ptr 对应的 MATLAB 阵列的第一个数据元素。如果 array_ptr 对应的数据类型为非双精度型数据，则返回数据自动转换为双精度型数据。

2. 字符串阵列操作函数

1）创建字符串阵列

创建字符阵列的方法有两种：一种是利用 C 语言字符串直接创建字符串阵列；另一种是先创建字符阵列，然后通过 mxGetChars 函数获取字符阵列的数据指针，并对字符数据进行操作。利用 C 语言字符串直接创建字符串阵列的函数为 mxCreateString 和 mxCreateCharMatrixFromStrings，直接创建字符阵列的函数为 mxCreateCharArray，具体如下。

```
mxArray *mxCreateString(const char *str);
```

函数功能：创建一维字符型阵列。

参数说明：str 用于初始化待创建 MATLAB 字符阵列的字符串。

返回值：创建的 MATLAB 字符类型阵列的 mxArray 指针。

```
mxArray *mxCreateCharMatrixFromString(mwSize m, const char **str);
```

函数功能：创建二维字符阵列，其初值为输入的 C 语言字符串数组。

参数说明：m 表示输入的 C 语言字符串的个数，即创建的字符矩阵的行数；str 表示输入的 C 语言字符串数组。

返回值：如果函数调用成功，则返回创建的二维字符阵列的 mxArray 指针，否则返回空。

附加说明：用这种方式创建的二维数组，由 m 指定行数，列数由输入的 m 个字符的最大长度确定。此外，在 MATLAB 字符阵列中，字符类型 mxChar，而不是 Char。通过例 11-21，大致可以了解 mxCreateCharMatrixFromStrings 函数的使用方法，其代码和执行结果如下。

【例 11-21】 mxCreateCharMatrixFromStrings 函数创建一个 MATLAB 字符阵列。

```
/*保存为 mxCreateCharMatrixFromStrings.c */
#include "mex.h"
#include "matrix.h"
void mexFunction(int nlhs, mxArray *plhs[], int nrhs,const mxArray *prhs[])
{
    char s1[] = "Bei Jing";
    char s2[] = "Shang Hai";
    char s3[] = "Tian Jin";
    char s4[] = "Chong Qing";
    char * s[4] = {s1,s2,s3,s4};
    mxArray * charArray;
    /*根据 C 语言字符串数组 s 创建字符阵列*/
    charArray = mxCreateCharMatrixFromStrings(4,s);
    if(nlhs==1)
    {
        plhs[0]=charArray;
    }
    else
      {
        mexPrintf("输出阵列的个数错误!\n");
        mxDestroyArray(charArray);
      }
}
```

执行结果如下。

```
>>str = mxCreateCharMatrixFromString;
>>str
str =
Bei Jing
Shang Har
Tian Jin
Chong Qing
>>
mxArray *mxCreateCharArray(mwSize ndim, const mwSize *dims);
```

函数功能：创建字符阵列。

参数说明：ndim 表示待创建字符阵列的维数，ndim 必须大于 0，由于 MATLAB 中标量用的阵列表示，因而 ndim 应该大于等于 2；dims 待创建字符阵列的各维大小，整型数组 dims 的维数应该大于等于 2。

返回值：返回创建的字符阵列，如果失败则返回 NULL。

2）字符串阵列转换为 C 语言字符串

字符串转化时 MEX 文件中操作字符串关键问题之一，MATLAB 提供了 mxArrayToString 和 mxGetString 两个可用于转换字符串的函数，这两个函数均可以将字符型阵列 mxArray 中的字符串转换为 C 语言可处理的字符串，其中，前者可以转换双字节字符集，后者只能转换单字节字符集。mxArrayToString 和 mxGetString 两个字符串转换函数

的详细说明如下。

```
char *mxArrayToString(const mxArray *array_ptr);
```

函数功能：将字符阵列转换为 C 语言字符串。

参数说明：array_ptr 待转换字符阵列。

返回值：返回转换后的字符串，如果函数调用失败则返回 NULL 空指针，此函数用以转换双字节的字符串。使用完毕后，用户应当释放 mxArrayToString 函数返回指针指向的内存块。

```
int mxGetString(const mxArray *array_ptr, char *buff, int buflen);
```

函数功能：得到 MATLAB 字符类型阵列的字符数据，此时得到的字符数据不是 MATLAB 的 mxChar 类型的，而是 C 语言 char 类型的，并且是以“\0”字符结束。

参数说明：array_ptr 是 MATLAB 字符阵列的 mxArray 指针；buff 是接受 MATLAB 字符阵列字符串的 C 语言字符串指针；bufflen 一维 buff 字符数组的大小，包括 C 语言字符串结束符‘\0’在内。

返回值：如果函数调用成功，并且 buff 的大小足够接受 MATLAB 字符阵列的所有字符，则返回 0，否则返回 1。

附加说明：如果输入的 MATLAB 字符阵列是多维的，此时将会按照一维的方式返回字符阵列中所有的字符，此时返回的字符串是 MATLAB 字符阵列按列存储的结果。如['ABCD';'EFGH']返回的是字符串“ABCDEFGH”。为了说明上述两个字符串转换函数的使用方法，编写了 mxGetStringEx.c 实例，该实例的主要功能即将输入的字符串通过 MEX 文件在命令行窗口中输入，通过如下命令将 mxGetStringEx.c 编译为 MEX 文件。

```
mex mxGetStringEx.c MyGetStringEx.c;
```

mxGetStringEx.c 的调用格式如下。

```
mxGetStringEx(str1,str2,…,nmode)或mxGetStringEx(str1)
```

其中，nmode 参数用于指定 mxGetStringEx 的工作模式，nmode = 1 采用 mxGetString 函数转换字符串，nmode = 2 采用 mxArrayToString 函数转换字符串。如【例 11-22】所示，mxGetStringEx 的测试结果和代码如下所示。

【例 11-22】 试将字符串阵列转换为 C 语言字符串。

```
/*保存为mxGetStringEx.c*/
#include "mex.h"
#include "matrix.h"
#ifndef NULL
#define NULL 0
#endif
char * MyGetString(mxArray * c,int nmode);
void mexFunction(int nlhs, mxArray *plhs[], int nrhs,
              const mxArray *prhs[])
{
int i=0;
```

```
    mwSize j=0;
    char * buff;
    int nmode;
    int nstr;
    nmode = (nrhs>1)?(mxGetScalar(prhs[nrhs-1])):(1);
    nstr = (nrhs>1)?(nrhs-1):(nrhs);
    for(i=0;i<nstr;i++)
    {
            if(!mxIsChar(prhs[i]))
            {
              mexPrintf("第%d 个输入的参数不是字符数组!\n",i+1);
              continue;
            }
            buff = MyGetString(prhs[i], nmode);
            mexPrintf("字符串%d:%s\n",i+1,buff);
            mxFree(buff);
            buff = NULL;
        }
    }
```

执行结果如下。

```
>> mxGetStringEx('字符串 1','字符串 2',1);
Warning: 字符串接收空间不足!
字符串 1: 字符
Warning: 字符串接收空间不足!
字符串 2: 字符
>> mxGetStringEx('字符串 1','字符串 2',2);
字符串 1: 字符串 1
字符串 2: 字符串 2
>>
```

3）获取字符阵列的数据

```
mxChar *mxGetChars(const mxArray *array_ptr);
```

函数功能：得到 MATLAB 字符类型阵列字符数据的指针。

输入参数：array_ptr 表示字符阵列 mxArray 的指针。

输出参数：array_ptr 对应字符阵列的字符数据的指针，由于 MATLAB 字符阵列中字符均以 mxChar 的方式存储，所以字符数据指针的类型为 mxChar *类型。

附加说明：mxGetChars 函数可以获取字符阵列字符数据的指针，通过此数据指针可以对 MATLAB 字符阵列进行修改、赋值等操作。

除上述函数外，还有逻辑性阵列操作函数、稀疏矩阵操作函数、结构体操作函数、元组阵列操作函数、类对象阵列属性操作函数、内存操作函数以及索引、维数和元素个数操作函数等其他操作函数，这里不一一赘述，读者可查阅相关资料了解。

【例 11-23】 编写 C 语言 MEX 程序代码，实现将输入量的 2 倍进行输出。

首先编写 C 语言程序代码，如下所示。

```
#include<math.h>
```

```
void timestwo(double y[],double x[])
{
y[0]=2.0*x[0]
return;
}
```

然后编写对应的 C 语言 MEX 程序代码，如下所示。

```
#include "mex.h"
void timestwo(double y[],double x[])
{
y[0]=2.0*x[0];
}
void mexFunction(int nlhs,mxArray *plhs[],int nrhs,const mxArray *prhs[])
{
  double *x,*y;
  int mrows,ncols;  /*检测输入参数个数*/

  if(nrhs!=1)
  {
    mexErrMsgTxt("One input required.");
}
else if(nlhs>1)
{
  mexErrMsgTxt("Too many output arguments");
/*确保输入参数是标量，且为正值*/
}
mrows=mxGetM(prhs[0]);
ncols=mxGetN(prhs[0]);
if
(!mxIsDouble(prhs[0])||mxIsComplex(prhs[0])||!(mrows==1&&ncols==1))
{
  mexErrMsgTxt("Input must be a noncomplex scalar double.");
/*创建返回参数的属性*/
}
plhs[0]=mxCreateDoubleMatrix(mrows,ncols,mxREAL);
/*为输入参数和输出参数分配指针*/
x=mxGetPr(prhs[0]);
y=mxGetPr(plhs[0]);
timestwo(y,x);      /*调用子函数*/
}
```

将上述文件保存为 timestwo.c，在 MATLAB 环境中进行编译，在命令窗口中输入如下语句。

```
mex timestwo.c   %编译代码
which timestwo.mexw32    %定位编译后的文件
```

最后得到编译后的文件信息为

```
C:\Users\admin\Documents\MATLAB\timestwo.mexw32
```

命令窗口中的输出结果如下。

```
x =
     2
y=timestwo(x)
y =
     4
```

11.6 MATLAB 与外部设备和互联网交互

MATLAB 之所以具备广泛的应用领域，具有十分强大的功能，一方面是由于 MATLAB 中具有能实现与外部设备进行通信的大量函数，包括串口、网络及即插即用等设备，另一方面还具备能够实现设备驱动程序编写的大量函数。

现在，通过两个简单的实例了解 MATLAB 这方面的强大功能。

【例 11-24】 举例说明 USB 接口设备拍摄的图像的读取。

在命令窗口中输入如下语句。

```
winvideoinfo=imaqhwinfo('winvideo')
device1=winvideoinfo.DeviceInfo(1)
device1.DefaultFormat
```

命令窗口中的输出结果如下。

```
winvideoinfo =
AdaptorDllName:'D:\MATLAB\R2009a\toolbox\imaq\imaqadaptors\win32\
mwwinvideoimaq.dll'
    AdaptorDllVersion: '3.3 (R2009a)'
          AdaptorName: 'winvideo'
            DeviceIDs: {[1]}
           DeviceInfo: [1x1 struct]

device1 =
          DefaultFormat: 'RGB24 640x480'
          DeviceFileSupported: 0
          DeviceName: 'Integrated Camera'
          DeviceID: 1
      ObjectConstructor: 'videoinput('winvideo', 1)'
       SupportedFormats: {1x16 cell}
ans =
RGB24_640x480
```

上述结果显示，系统安装一个 winvideo 设备，该设备是 Integrated Camera 设备，默认图像格式为 RGB24_640x480。

其次在命令窗口中输入如下语句。

```
clear
clc
```

```
obj=videoinput('winvideo',1);
preview(obj);
```

在屏幕上则出现如图 11-26 所示的摄像头的连续图像。

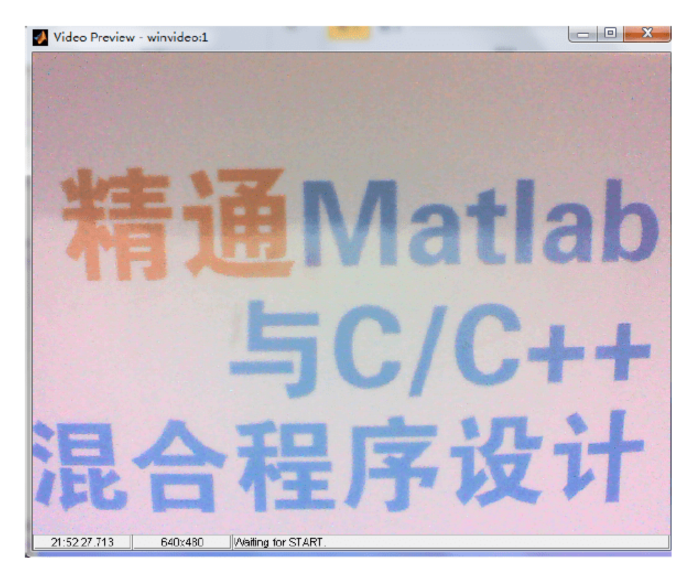


图 11-26 截屏结果

【例 11-25】 举例说明 MATLAB 与互联网的交互功能。

输入如下所示的代码并保存。

```
t=tcpip('www.chemanman.com',80);      %设置连接网站及连接端口
set(t,'InputBufferSize',5000);        %设置连接的缓冲区
fopen(t)                              %打开连接
fprintf(t,'GET/');                    %得到首页内容
get(t,'BytesAvailable')               %显示可用字节数
A=[];                                 %设置初值
while(get(t,'BytesAvailable')>0)      %如果剩余字节数不为 0
    A=[A,fscanf(t),13];               %fscanf(t)按行读取数据，并且文件指针移动
                                      %13 为回车键对应的 ASCII 码
end
if isempty(A)
    disp('No data');
else
    A=A       %显示读取结果
end
fclose(t);  %关闭连接
delete(t);  %删除连接
clear t     %清除连接
```

执行结果等效于打开指定网站主页后，单击页面菜单中“查看源代码”选项的结果。

11.7 本章小结

本章主要介绍 MATLAB 提供与外部交互的强大功能。MATLAB 的外部接口使得

MATLAB 可以与外部设备和程序实现数据交互和程序移植，可以扩充 MATLAB 强大的数值计算和图形显示功能。通过 MATLAB 接口编程，可以充分利用现有资源，能更容易地编写出功能强大、结构简洁的应用程序。本章着重介绍了如何在 Word 和 Excel 中使用 MATLAB 资源，在 C 语言中调用 MATLAB 资源，调用外部设备及互联网资源等内容。

11.8 习题

（1）如何配置 MEX 文件编译器？C 语言的 MEX 文件由哪几部分组成？

（2）创建一个 M-book 文件，输入以下文字。

“clear

输入矩阵 c: c = [1 2; 3 4; 5 6]”

将“clear”定义为自动初始化单元，将“c = [1 2;3 4;5 6]”定义为输入单元，并得出输出单元。

（3）编写 C 语言 MEX 程序代码，实现将输入量的 4 倍进行输出。